Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Veronika Horvat

Matrične faktorizacije nad polkolobarji

magistrsko delo

študijski program druge stopnje
računalništvo in informatika

Mentorica: izr. prof. dr. Polona Oblak

Ljubljana, 2017

Zahvala

Rada bi se zahvalila mentorici Poloni Oblak za pomoč pri izbiri teme, hitro odzivnost in spodbudo, ko sem jo najbolj potrebovala. Hvala podjetju SODO za podatke, s katerimi sem lahko izvedla praktični del naloge. Iskrena hvala tudi moji družini, sošolcem in Kaji za vso podporo tekom študijskih let.

Veronika Horvat, 2017

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CSV	Comma-separated values	Podatki, ločeni z vejico
TXT	Text file	Tekstovna datoteka
FMF	Faculty of Mathematics and Physics	Fakulteta za matematiko in fiziko
FRI	Faculty of Computer and Information Science	Fakulteta za računalništvo in informatiko
SVD	Singular Value Decomposition	Singularni razcep
NMF	Non-negative matrix factorization	Nenegativna matrična faktorizacija

Povzetek

Naslov: Matrične faktorizacije nad polkolobarji

Polkolobar je algebraična skruktura, s katero je med drugim mogoče nekatere probleme, ki so v klasični algebri rešljivi na nelinearen način, rešiti na linearen način. Z matričnimi faktorizacijami na dve ali več matrik, je mogoče rešiti različne probleme v podatkovnem rudarjenju in strojnem učenju. V delu predstavimo reševanje problema iskanja lastnosti izrazno-dokumentnih matrik ter napovedovanja proizvodnje električne energije obnovljivih virov s pomočjo matrične faktorizacije nad polkolobarji. Opišemo delovanje algoritma Cancer, enega izmed algoritmov matrične faktorizacije nad različnimi polkolobarji ter kako vsak izmed njih vpliva na rezultate algoritma.

Ključne besede

polkolobar, matrična faktorizacija, algoritem Cancer, napoved proizvodnje električne energije, izrazno-dokumentna matrika

Abstract

Title: Matrix factorization over semirings

Semiring is an algebraic structure with which some problems that are nonlinear in classical algebra, can be solved in a linear manner. By matrix factorization, decomposing input matrix into two (or more) matrices, various problems in data mining and machine learning can be solved. We present how the problems of finding properties of term-document matrices and forecasting electricity production of renewable resources can be solved by matrix factorization over semirings. We describe algorithm Cancer, one the algorithms for matrix factorization over different semirings and compare how each of them affects the results.

Keywords

semiring, matrix factorization, algorithm Cancer, forecast of electrical energy, term-document matrix

Poglavje 1

Uvod

V magistrskem delu se bomo nekoliko oddaljili od klasičnih algebraičnih struktur kot so grupe, kolobarji in obsegi ter spoznali strukture, ki so v nekaterih primerih primernejše za modeliranje in reševanje določenih problemov [7]. Eden izmed teh je dobro poznan problem iskanja najcenejših poti v grafu, katerega lahko v klasični algebri rešimo z uporabo nelinearnega sistema Bellmanovih enačb. Enačbe lahko zapišemo kot linearen sistem, če zamenjamo + in × s primernima operacijama. V kolikor ti dve operaciji zadoščata predpisanim lastnostim, novo strukturo imenujemo polkolobar. Polkolobarji omogočajo generično reševanje problemov in zmanjšujejo problem kompleksnosti. Poznanih je več vrst, s pomočjo katerih rešujemo različne probleme, kot so preverjanje obstoja poti v povezanem grafu, iskanje najkrajše oziroma najdaljše poti, računanje največje možne kapacitete poti, zanesljivosti omrežja, verjetnosti obstoja omrežij...

Večina omenjenih problemov je grafološke narave, zato smo želeli poiskali primere uporabe polkolobarjev za reševanje negrafoloških problemov. Osredotočili smo se na matrične faktorizacije, pri katerih gre za dekompozicijo vhodne matrike na dve ali več matrik. Navadno je glavni cilj faktorizacije pridobiti matrike nizkih redov, katerih produkt je karseda najboljša aproksimacija vhodne matrike.

Verjetno najbolj znana primera matrične faktorizacije sta singularni raz-

cep (SVD) in nenegativna matrična faktorizacija (NMF). Obe faktorizaciji sta prisotni v podatkovnem rudarjenju in strojnem učenju. Primeri uporabe so procesiranje signalov, kjer pridobimo informacije o obnašanju nekega dogodka, prepoznavanje vzorcev v podatkih ter iskanje manjkajočih vrednosti v matrikah. Iskanje manjkajočih vrednosti v matrikah je v praksi uporabno v priporočilnih sistemih in jih denimo določimo nenegativno matrično faktorizacijo nad aritmetičnim polkolobarjem. Matriko, sestavljeno iz uporabnikov in izdelkov, katere elementi predstavljajo ocene uporabnika za nek izdelek, z matrično faktorizacijo dekomponiramo. S pomočjo te pridobimo manjkajoče elemente oziroma predvidevane ocene izdelkov glede na preference uporabnika, na podlagi katerih lahko uporabnikom predstavimo priporočila.

Drugi primer matrične faktorizacije je Boolova matrična faktorizacija [15], ki deluje nad Boolovim polkolobarjem. Uporablja se v podatkovnem rudarjenju, kombinatoriki, za iskanje vlog, na podlagi katerih je osebam dodeljen dostop do nekega sistema in drugih.

Faktorizacijska algoritma, ki iščeta vzorce iz podatkov sta Capricorn in Cancer, ki se izvajata nad max-krat polkolobarjem. Uporabljata se pri iskanju robov na slikah. Produkt izhodnih matrik, ki jih pridobimo z izvajanjem enega izmed algoritmov lahko zapišemo kot vsoto matrik ranga ena, ki predstavljajo enostavne vzorce iz podatkov.

V okviru magistrskega dela smo se odločili implementirati enega izmed algoritmov matrične faktorizacije in ga uporabiti za reševanje novih problemov, kot sta interpretiranje izrazno-dokumentnih matrik in napovedovanje proizvodnje električne energije obnovljivih virov. Ker so v podatkih prevladovala realna števila, smo se odločili za uporabo algoritma Cancer, ki je nad takšnimi podatki prinesel najboljše rezultate. Za razliko od algoritma Cancer je algoritem Capricorn bolj primeren za manipulacijo z diskretnimi podatki. Implementaciji obeh algoritmov sta sicer javno dostopni, vendar smo se odločili, da algoritem Cancer za boljše razumevanje sami implementiramo. S tem smo pridobili tudi enostavno nadgradnjo implementacije, saj smo želeli imeti možnost uporabe algoritma nad različnimi vrstami polkolo-

barjev in ne samo nad max-krat polkolobarjem.

Delovanje implementiranega algoritma smo preverili na dveh vrstah problemov. Za reševanje prvega problema smo implementirali program, ki iz tekstovnih datotek izlušči uporabne izraze in tvori izrazno-dokumentrno matriko. Z uporabo algoritma Cancer smo ugotovili zanimivosti besedil in izrazov. Drugi problem je napovedovanje proizvodnje električne energije obnovljivih virov s pomočjo podatkov, pridobljenih od podjetja SODO. Izračunane napovedi matrične faktorizacije smo primerjali z napovedmi podjetja SODO ter dejanskimi proizvodnjami.

Magistrska naloga je sestavljena iz štirih glavnih poglavij. V poglavju 2 predstavimo teorijo polkolobarjev ter njihovo povezavo z matrikami in grafi. V poglavju 3 je predstavljen pojem matrične faktorizacije in primeri njene uporabe. Opisan je tudi eden izmed algoritmov matrične faktorizacije, algoritem Cancer. V 4. poglavju opišemo problem izrazno-dokumentnih matrik in prikažemo meritve izvajanja matrične faktorizacije z opisi študijskih predmetov. V zadnjem, 5. poglavju, pa predstavimo reševanje problema napovedovanja proizvodnje električne energije ter opišemo rezultate in meritve.

Poglavje 2

Polkolobar

V poglavju bomo predstavili polkolobar, glavno strukturo, ki smo jo uporabili za reševanje naših problemov. Začeli bomo z njegovo definicijo ter osnovnimi lastnostmi. Za tem bomo opisali povezavo polkolobarjev z matrikami in matrikam pripadajočimi grafi ter opisali njihove lastnosti. Na koncu bomo našteli nekaj najbolj znanih vrst polkolobarjev in probleme, ki jih ti rešujejo.

2.1 Definicija

Definicija 2.1 *Polkolobar* $(S, \oplus, \otimes)$ *je neprazna množica elementov* S*, na kateri sta definirani operaciji seštevanja* $\oplus$ *in množenja* $\otimes$ *ter nevtralna elementa* ϵ *in* e *z naslednjimi lastnostmi.*

1. Operacija seštevanja je komutativna:

$$x \oplus y = y \oplus x \qquad \forall x, y \in S \tag{2.1}$$

2. Operacija seštevanja je asociativna:

$$(x \oplus y) \oplus z = x \oplus (y \oplus z) \qquad \forall x, y, z \in S \tag{2.2}$$

3. Operacija množenja je asociativna:

$$(x \otimes y) \otimes z = x \otimes (y \otimes z) \qquad \forall x, y, z \in S$$

4. Operacija množenja je distributivna nad operacijo seštevanja:

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z) \qquad \forall x, y, z \in S$$
$$(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z) \qquad \forall x, y, z \in S$$

5. Obstaja nevtralni element seštevanja ϵ:

$$x \oplus \epsilon = \epsilon \oplus x = x \qquad \forall x \in S$$

6. Obstaja nevtralni element množenja e:

$$x \otimes e = e \otimes x = x \qquad \forall x \in S$$

7. ϵ je absorbirajoč element množenja:

$$x \otimes \epsilon = \epsilon \otimes x = \epsilon \qquad \forall x \in S$$

Polkolobar je komutativen, če je $\otimes$ komutativna operacija:

$$x \otimes y = y \otimes x \qquad \forall x, y \in S$$

Poglejmo si, kako operiramo z matrikami, katere elementi so elementi nekega polkolobarja. Naj bo $(S, \oplus, \otimes)$ polkolobar ter $M_n(S)$ množica matrik velikosti $n \times n$ z elementi iz množice S. Iz $M_n(S)$ izberemo matriki $A = (a_{ij})$ in $B = (b_{ij})$. Vsota matrik A in B, $A \oplus B$, je enaka matriki $C = (c_{ij}) \in M_n(S)$, kjer je:

$$c_{ij} = a_{ij} \oplus b_{ij},$$

za vsak $i, j = 1, \ldots, n$.

Primer 2.1 *V polkolobarju* $([0, \infty], \min, \times)$ *je vsota matrik.*

$$A = \begin{bmatrix} 2 & \infty \\ 1 & 3 \end{bmatrix} \quad in \quad B = \begin{bmatrix} 5 & 2 \\ 5 & 3 \end{bmatrix}$$

enaka

$$A \oplus B = \begin{bmatrix} \min\{2,5\} & \min\{\infty,2\} \\ \min\{1,5\} & \min\{3,3\} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}.$$

Produkt matrik A in B, $A \otimes B$, je enak matriki $D = (d_{ij}) \in M_n(S)$, kjer je

$$d_{ij} = \bigoplus_{k=1}^{n} a_{ik} \otimes b_{kj},$$

za vsak $i, j = 1, \ldots, n$. Produkt bomo označevali kot AB ali $A \otimes B$.

Primer 2.2 *V polkolobarju* $([0, \infty], \min, \times)$ *zmnožimo matriki A in B iz primera 2.1, kot*

$$A \otimes B = \begin{bmatrix} \min\{2 \times 5, \infty \times 5\} & \min\{2 \times 2, \infty \times 3\} \\ \min\{1 \times 5, 3 \times 5\} & \min\{1 \times 2, 3 \times 3\} \end{bmatrix} =$$

$$= \begin{bmatrix} \min\{10, \infty\} & \min\{4, \infty\} \\ \min\{5, 15\} & \min\{2, 9\} \end{bmatrix} = \begin{bmatrix} 10 & 4 \\ 5 & 2 \end{bmatrix}.$$

Če je v polkolobarju S element ϵ nevtralni element za seštevanje, e pa nevtralni element za množenje, je matrika $\sum = \begin{bmatrix} \epsilon & \epsilon & \cdots & \epsilon \\ \epsilon & \epsilon & \cdots & \epsilon \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon & \epsilon & \cdots & \epsilon \end{bmatrix}$ nevtralni element seštevanja množice matrik $M_n(S)$, matrika $I = \begin{bmatrix} e & \epsilon & \epsilon & \cdots & \epsilon \\ \epsilon & e & \epsilon & \cdots & \epsilon \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \epsilon & \epsilon & \cdots & \epsilon & e \end{bmatrix}$ pa nevtralni element za množenje v $M_n(S)$.

Ker je seštevanje matrik definirano po komponentah, je operacija seštevanja matrik asociativna in komutativna. Asociativnost množenja in distributivnost pa je mogoče preveriti s krajšim računom [10]. S temi lastnostmi množica matrik $M_n(S)$ postane polkolobar. Zaradi asociativnosti množenja matrik lahko definiramo potence matrik kot

$$A^0 = I$$
$$A^k = A^{k-1} \otimes A$$

za $k \geq 1$.

Za vsak $k \in \mathbb{N}$ lahko definiramo tudi matriko $A^{(k)} = I \oplus A \oplus A^2 \oplus \cdots \oplus A^k$. Če obstaja limita $A^{(k)}$, obstaja element

$$A^* = \lim_{k \to \infty} A^{(k)},$$

ki ga imenujemo kvazi inverz matrike A.

Če je $A^{(m-1)} = A^{(m)}$ za nek $m \in \mathbb{N}$, potem je

$$\begin{aligned} A^{(m+1)} &= I \oplus A \oplus \cdots \oplus A^m \oplus A^{m+1} = \\ &= I \oplus A \otimes \underbrace{(I \oplus \cdots \oplus A^m)}_{A^{(m)}} = \\ &= I \oplus A \otimes \underbrace{(I \oplus \cdots \oplus A^{m-1})}_{A^{(m-1)}} = \\ &= I \oplus A \oplus A^2 \oplus \cdots \oplus A^m = A^{(m)} = A^{(m-1)} \end{aligned}$$

in zato je $A^{(m+k)} = A^{(m-1)}$ za vse $k \geq 0$, zatorej $A^* = A^{(m-1)}$.

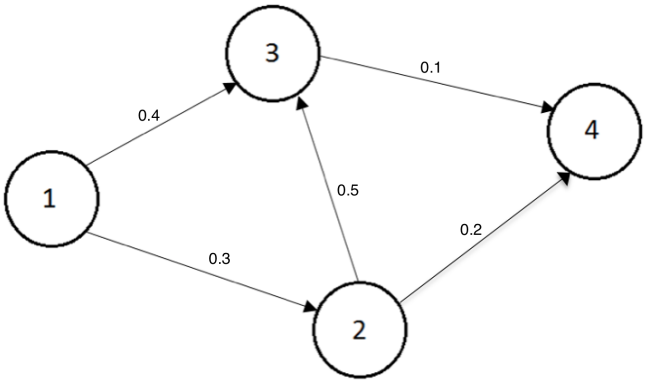
2.2 Povezava z grafi

Polkolobarji rešujejo veliko problemov v povezavi z grafi, zato si poglejmo njihove osnovne lastnosti in primere uporabe.

Definicija 2.2 *Usmerjen graf G je končna množica vozlišč $V = V(G) = \{1, 2, \ldots, n\}$ in množica usmerjenih povezav $E = E(G) \subseteq V \times V$. To pomeni, da (i, j) označuje povezavo iz vozlišča i v vozliče j. Utežen graf je graf, v katerem vsaki povezavi (i, j) priredimo uteži w_{ij}. Sprehod v grafu G iz vozlišča i v vozlišče j je zaporedje vozlišč $i = x_0, x_1, \cdots, x_{k-1}, x_k = j$, kjer k imenujemo dolžina sprehoda. V uteženem grafu je utež sprehoda enaka produktu uteži na povezavah sprehoda. Omejenemu sprehodu, kjer so vse točke različne rečemo pot.*

Osredotočili se bomo na grafe v katerih med dvema vozliščema obstaja največ ena povezava.

Vsak usmerjen utežen graf G z n vozlišči lahko zapišemo z matriko A velikosti $n \times n$ in obratno. Vsak element matrike A predstavlja ceno povezave med dvema vozliščema, torej element a_{ij} je cena povezave iz vozlišča i v vozlišče j. V primeru, da povezava med dvema vozliščema ne obstaja, je element v matriki enak ϵ. Več si lahko o povezavah med grafi in matrikami preberemo v [6, 3].



Slika 2.1: Usmerjen utežen graf G s štirimi vozlišči.

Primer 2.3 *Graf G iz slike 2.1 pripada matriki* $A = \begin{bmatrix} \epsilon & 0.3 & 0.4 & \epsilon \\ \epsilon & \epsilon & 0.5 & 0.2 \\ \epsilon & \epsilon & \epsilon & 0.1 \\ \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}$.

Trditev 2.1 *Če graf G z vozlišči $1, 2, \ldots, n$ pripada matriki A, je (i, j)-ti element matrike A^k skupna utež sprehodov dolžine k iz i v j, za vsak $k \in \{1, 2, \ldots, n\}$.*

Dokaz. *Ta trditev je enostavno dokazljiva z indukcijo.*

1. *Za $k = 1$ je $A^1 = A$ in a_{ij} je utež povezave med i in j.*

2. *Predvidevamo, da trditev velja za k. Dokazati želimo, da velja tudi za $k + 1$. Naj bo b_{ij} (i, j)-ti element A^k in a_{ij} (i, j)-ti element matrike*

A. Ker smo predpostavili, da trditev velja za število k, b_{ij} predstavlja skupno utež sprehodov dolžine k od i do j. Po definiciji je

$$(A^{k+1})_{ij} = (A^k A)_{ij} = b_{i1} \otimes a_{1j} \oplus b_{i2} \otimes a_{2j} \oplus \cdots \oplus b_{in} \otimes a_{nj} = \bigoplus_{m=1}^{n} b_{im} \otimes a_{mj} \tag{2.3}$$

Ker je $b_{i1} \otimes a_{1j}$ zmnožek skupne uteži sprehodov dolžine k iz i v 1 in uteži povezave iz 1 v j, je zmnožek enak skupni uteži sprehodov dolžine $k+1$ iz i v j, kjer je 1 predzadnje vozlišče. Podobno za vsak m velja, da je $b_{im} \otimes a_{mj}$ skupna utež sprehodov iz i v j, v katerem je m predzadnje vozlišče.
Seštevek (2.3) je torej skupna utež vseh možnih poti dolžine $k+1$ iz i v j.

□

Posledica 2.1 *Če graf G s točkami $1, 2, \ldots, n$ pripada matriki A, je za vsak $k \in \{1, 2, \ldots, n\}$, (i, j)-ti element matrike $A^{(k)}$ skupna utež sprehodov dolžine največ k iz i v j.*

Posledica 2.2 *Če graf G s točkami $1, 2, \ldots, n$ pripada matriki A in obstaja A^*, je (i, j)-ti element matrike A^* skupna utež sprehodov iz i v j.*

Posledici 2.1 in 2.2 je mogoče podobno kot trditev 2.1 dokazati z indukcijo.

2.3 Vrste polkolobarjev

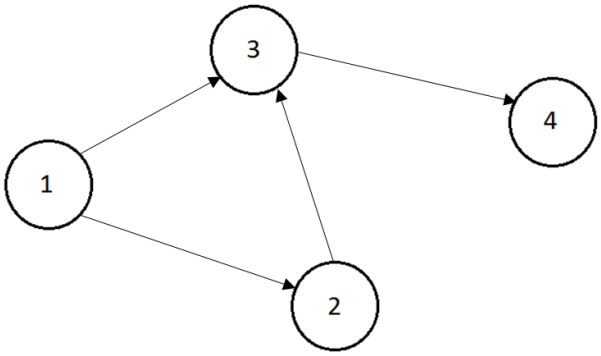
V naslednjih podpoglavjih bomo spoznali nekaj vrst polkolobarjev, njihove lastnosti ter probleme, ki jih rešujejo. Prepričati se, da za vsakega izmed omenjenih polkolobarjev veljajo vse potrebne lastnosti polkolobarjev ni pretežko, vendar zelo tehnično, zato bomo dokaze izpustili.

2.3.1 Boolov polkolobar

Boolov polkolobar je najenostavnejši netrivialni polkolobar in je definiran kot:

$$S = \{0, 1\}, \quad \oplus = \vee, \quad \otimes = \wedge, \quad \epsilon = 0 \quad \text{in} \quad e = 1.$$

Z njim je mogoče rešiti problem povezljivosti v grafu. Če ga pretvorimo v pripadajočo matriko A, na mesta kjer povezava obstaja, podamo število 1, na ostala pa 0. Obstoj sprehoda iz vozlišča i v vozlišče j dolžine n izračunamo tako, da matriko A potenciramo na n-to potenco, tako je utež sprehoda enaka produktu uteži povezav na tem sprehodu. V primeru, da je po potenciranju na mestu (i, j) število 1, povezava obstaja. V nasprotnem primeru je na istem mestu število 0. Poglejmo si primer.



Slika 2.2: Usmerjen graf G s štirimi vozlišči.

Primer 2.4 *Grafu G s slike 2.2 pripada matrika*

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Želimo preveriti, če obstaja sprehod dolžine 2 iz prvega v četrto vozlišče. Izračunamo $(A^2)_{1,4} = \vee\{1 \wedge 0, 1 \wedge 0, 1 \wedge 1, 0 \wedge 1\} = \vee\{0, 0, 1, 0\} = 1$*, ali splošneje*

$$A^2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Iz matrike A^2 *vidimo, da sprehod iz prvega v četrto vozlišče obstaja. Ne samo to, obstajata še dva sprehoda dolžine 2. Iz prvega v tretje in iz drugega v četrto vozlišče.*

2.3.2 Polkolobar ozkega grla

Polkolobar ozkega grla je definiran kot:

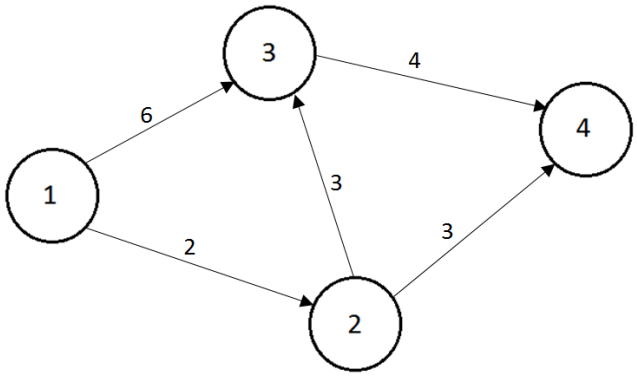
$$S = \mathbb{R}_+ \cup \{\infty\}, \quad \oplus = \max, \quad \otimes = \min, \quad \epsilon = 0 \quad \text{in} \quad e = \infty.$$

Polkolobar, ki mu v angleščini rečemo "bottleneck semiring", rešuje problem maksimalne kapacitete sprehoda v grafu. Tako kot v razdelku 2.3.1, tudi tukaj zapišemo informacije grafa v matriko A. Grafi so v tem primeru uteženi in to tako, da je za vsako povezavo med točkama i in j podana največja možna kapaciteta, ki je lahko poslana po tej povezavi. Želimo poiskati največjo možno kapaciteto, ki jo bo mogoče prenesti po sprehodu dolžine n iz točke i do j.

Primer 2.5 *Graf (slika 2.3) predstavimo v matrični obliki kot*

$$A = \begin{bmatrix} 0 & 2 & 6 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Želimo izvedeti, kakšna je največja možna kapaciteta, ki jo je mogoče prenesti po sprehodu dolžine 2 iz prvega do četrtega vozlišča. Izračunamo $(A^2)_{1,4} =$



Slika 2.3: Usmerjen utežen graf G s štirimi vozlišči.

$\max\{\min\{0,0\},\min\{2,3\},\min\{6,4\},\min\{0,0\}\} = \max\{0,2,4,0\} = 4$, *oziroma splošneje*

$$A^2 = \begin{bmatrix} 0 & 2 & 6 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 2 & 6 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 6 & 4 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

Rešitev je element v matriki, ki se nahaja v prvi vrstici in četrtem stolpcu. Največja možna kapaciteta je 4.

2.3.3 Aritmetični polkolobar

Dobro poznan aritmetični polkolobar je definiran kot

$$S = \mathbb{N}, \quad \oplus = +, \quad \otimes = \times, \quad \epsilon = 0 \quad \text{in} \quad e = 1.$$

V svetu grafov prinaša pozitivne rezultate pri iskanju števila sprehodov v grafu oziroma po koliko različnih sprehodih lahko pridemo iz točke i v točko j. Tako kot pri prejšnjih primerih, tudi tokrat graf najprej zapišemo v matriko ter nato operiramo z njo.

2.3.4 Max-krat in verjetnostni polkolobar

Max-krat polkolobar, ki ga bomo v podpoglavju 3.2 uporabili za razlago algoritma Cancer, je definiran kot

$$S = \mathbb{R}_+, \quad \oplus = \max, \quad \otimes = \times, \quad \epsilon = 0 \quad \text{in} \quad e = 1.$$

Max-krat polkolobarju precej podoben je verjetnostni polkolobar, ki je definiran kot

$$S = [0,1], \quad \oplus = \max, \quad \otimes = \times, \quad \epsilon = 0 \quad \text{in} \quad e = 1.$$

Dober primer uporabe je računanje največje zanesljivosti omrežij, kar lahko enačimo z računanjem maksimalne verjetnosti obstoja poti v grafu z disjunktnimi potmi. V primeru, da poti niso disjunktne, operacijo seštevanja zamenjamo s simetrično razliko ($a \oplus b = a + b - ab$) in tako dobimo polkolobar

$$S = [0,1], \quad \oplus = a + b - ab, \quad \otimes = \times, \quad \epsilon = 0 \quad \text{in} \quad e = 1, \tag{2.4}$$

ki ga uporabimo za izračun obstoja omrežij.

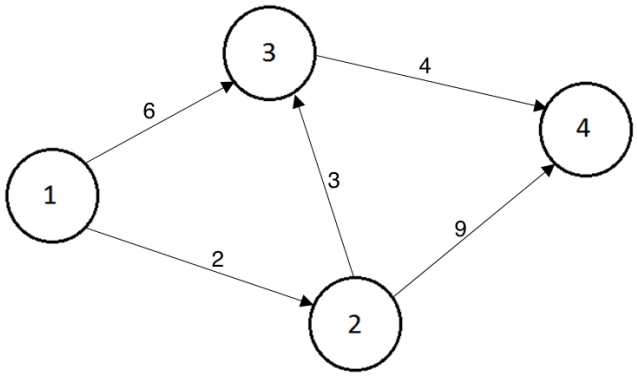
Primer 2.6 *Poglejmo graf na sliki 2.1 in njegovo pripadajočo matriko*

$$A = \begin{bmatrix} 0 & 0.3 & 0.4 & 0 \\ 0 & 0 & 0.5 & 0.2 \\ 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Zanima nas, kolikšna je verjetnost obstoja poti dolžine 2 iz prvega v četrto vozlišče. Tako kot pri vseh prejšnjih primerih izračunamo drugo potenco matrike A nad polkolobarjem (2.4)*. Zanima nas element* $(A^2)_{1,4} = (0 \otimes 0) \oplus (0.3 \otimes 0.2) \oplus (0.4 \otimes 0.1) \oplus (0 \otimes 0) = 0 \oplus 0.06 \oplus 0.04 \oplus 0 = 0.06 + 0.04 - 0.06 \times 0.04 = 0.098$*, ali splošneje*

$$A^2 = \begin{bmatrix} 0 & 0.3 & 0.4 & 0 \\ 0 & 0 & 0.5 & 0.2 \\ 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0.3 & 0.4 & 0 \\ 0 & 0 & 0.5 & 0.2 \\ 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.15 & 0.098 \\ 0 & 0 & 0 & 0.05 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Torej 0.098 je verjetnost obstoja poti dolžine 2 iz vozlišča 1 v vozlišče 4.



Slika 2.4: Usmerjen utežen graf G s štirimi vozlišči.

2.3.5 Tropski polkolobar in max-plus polkolobar

Tropski polkolobar je množica realnih števil z dodatnim elementom pozitivne neskončnosti, definirana kot:

$$S = \mathbb{R} \cup \{\infty\}, \quad \oplus = \min, \quad \otimes = +, \quad \epsilon = \infty \quad \text{in} \quad e = 0.$$

Uspešno rešuje problem iskanja najcenejših poti v grafu. Podobno kot v razdelkih 2.3.1 in 2.3.2, se problema lotimo tako, da grafu najprej zapišemo pripadajočo matriko z elementi, ki predstavljajo cene oziroma uteži povezav. Če želimo izvedeti, kateri je najcenejši sprehod dolžine n iz i-te v j-to točko, izračunamo n-to potenco matrike grafa. Poglejmo graf na sliki 2.4. Zanima nas torej najmanjša cena sprehoda dolžine 2 iz prvega v zadnje vozlišče. Pripadajoča matrika je enaka:

$$A = \begin{bmatrix} \infty & 2 & 6 & \infty \\ \infty & \infty & 3 & 9 \\ \infty & \infty & \infty & 4 \\ \infty & \infty & \infty & \infty \end{bmatrix}.$$

Ko izračunamo matriko A^2, na mestu $(A^2)_{1,4}$ najdemo rešitev

$$(A^2)_{1,4} = \min\{\infty + \infty, 2 + 9, 6 + 4, \infty + \infty\} = \min\{\infty, 11, 10, \infty\} = 10.$$

Včasih nas bolj zanima najcenejši sprehod iz vozlišča i do vozlišča j ne glede na dolžino. To izračunamo s pomočjo kvazi inverza $A^* = I \oplus A \oplus A^2 \oplus A^3 \oplus A^4 \oplus \cdots$, ki smo ga definirali na koncu podpoglavja 2.1. Matrika A predstavlja neposredne povezave med vozlišči, matrika A^2 predstavlja sprehode dolžine 2, matrika A^3 predstavlja sprehode dolžine 3 itd. Ker smo v polkolobarju, kjer je operacija seštevanja zamenjana z minimum med števili, po izračunu vsot med vsemi matrikami dobimo najmanjše elemente oziroma najcenejše sprehode izmed vseh, ne glede na dolžino.

Primer 2.7 *Zanima nas, kako v grafu iz slike 2.4 najceneje pridemo iz prve v četrto točko. Najprej zapišemo matriko A, nato pa izračunamo kvazi inverz iz katerega lahko razberemo rešitev.*

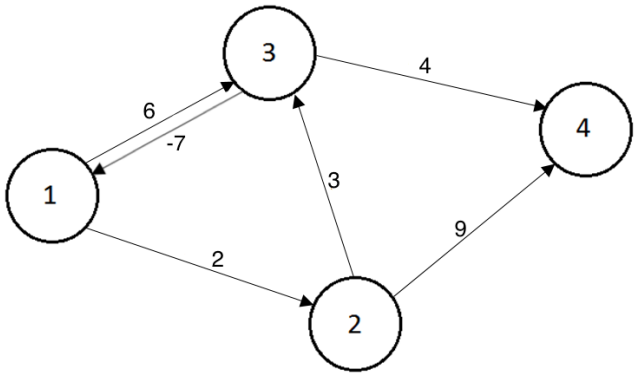
$$A^* = I \oplus A \oplus A^2 \oplus A^3 \oplus A^4 \oplus \cdots =$$

$$= \begin{bmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{bmatrix} \oplus \begin{bmatrix} \infty & 2 & 6 & \infty \\ \infty & \infty & 3 & 9 \\ \infty & \infty & \infty & 4 \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \begin{bmatrix} \infty & \infty & 5 & 10 \\ \infty & \infty & \infty & 7 \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus$$

$$\oplus \begin{bmatrix} \infty & \infty & \infty & 9 \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \cdots = \begin{bmatrix} 0 & 2 & 5 & 9 \\ \infty & 0 & 3 & 7 \\ \infty & \infty & 0 & 4 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

Iz prve vrstice dobljenega kvazi inverza lahko razberemo najcenejše sprehode iz prvega v vsa ostala vozlišča podanega grafa.

Primer 2.8 *Poglejmo si še en primer grafa na sliki 2.5 in izračunajmo najcenejše sprehode iz prvega vozlišča do vseh ostalih vozlišč. Tako kot v*



Slika 2.5: Usmerjen utežen graf G s štirimi vozlišči.

prejšnjem primeru bomo izračunali A^*.

$$A^* = \begin{bmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{bmatrix} \oplus \begin{bmatrix} \infty & 2 & 6 & \infty \\ \infty & \infty & 3 & 9 \\ -7 & \infty & \infty & 4 \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \begin{bmatrix} -1 & \infty & 5 & 10 \\ -4 & \infty & \infty & 7 \\ \infty & -5 & -1 & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus$$

$$\oplus \begin{bmatrix} -2 & 0 & 4 & 9 \\ -5 & \infty & 1 & 6 \\ -9 & -6 & -2 & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \begin{bmatrix} -3 & 0 & 3 & 8 \\ -6 & -3 & 1 & 5 \\ -9 & -7 & -3 & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \begin{bmatrix} -4 & -1 & 3 & \infty \\ -6 & -4 & 0 & 5 \\ -10 & -7 & -4 & 1 \\ \infty & \infty & \infty & \infty \end{bmatrix} \oplus \cdots$$

Vidimo, da izračun A^* *ni mogoč, saj graf vsebuje negativno utež na povezavi iz vozlišča 3 v vozlišče 1 in posledično cikel z negativno utežjo. Zaradi tega, so elementi z vsako višjo potenco matrike* A *manjši.*

Splošneje velja, če v grafu G z n vozlišči s pripadajočo matriko A ni ciklov z negativno utežjo, ima zaporedje matrik $A^{(k)}$ limito A^* in velja

$$A^* = \lim_{k \to \infty} A^{(k)} = A^{(n-1)} = A^{(n)} = \cdots$$

Dokaz lahko najdemo v [10].

Kvazi inverz je le eden od načinov, kako izračunati cene najcenejšega sprehoda v grafu. Isti problem lahko rešimo s pomočjo Bellmanovih enačb,

ki jih v tropskem polkolobarju lahko iterativno definiramo kot

$$y^1 = b^T$$
$$y^{t+1} = y^t \otimes A \oplus b^T,$$

kjer je $t = 1, 2, \ldots$ Pri tem je b^T vrstica dolžine n, s katero podamo vozlišče iz katerega želimo izračunati najcenejše sprehode. V vektorju b^T na mesto, za katerega želimo izračunati najcenejše sprehode, zapišemo e, na ostala pa ϵ. Če torej želimo izračunati najcenejše sprehode grafa na sliki 2.4 iz prvega vozlišča, je $b^T = \begin{bmatrix} 0 & \infty & \infty & \infty \end{bmatrix}$.

Ker v primeru 2.7 graf ne vsebuje ciklov z negativno utežjo, kvazi inverz izračunamo tako, da zmnožimo matriko do 3. potence, saj od tam dalje potence matrike postanejo stabilne in se ne spreminjajo. V primeru 2.8, kjer graf vsebuje cikel z negativno utežjo, pa kvazi inverz ne obstaja.

Podobno kot tropski je definiran max-plus polkolobar:

$$S = \mathbb{R} \cup \{-\infty\}, \quad \oplus = \max, \quad \otimes = +, \quad \epsilon = -\infty \quad \text{in} \quad e = 0,$$

s katerim rešujemo problem iskanja najdaljšega sprehoda v grafu.

Poglavje 3

Matrična faktorizacija

V tem poglavju bomo opisali matrično faktorizacijo, njene lastnosti ter probleme, ki jih lahko rešimo z uporabo njenih algoritmov nad določeno vrsto polkolobarja. V nadaljevanju bomo opisali implementacijo in delovanje enega izmed algoritmov matrične faktorizacije, algoritma Cancer, ki smo ga, nekoliko nadgrajenega, uporabili pri reševanju naših problemov.

3.1 Uvod

Matrična faktorizacija je množica algoritmov, katerih glavni cilj je aproksimacija matrike A velikosti $m \times n$ kot produkt dveh morda manjših matrik B in C, kjer je B velikosti $m \times k$ in C velikosti $k \times n$. Zapišemo jo kot

$$A \approx B \times C.$$

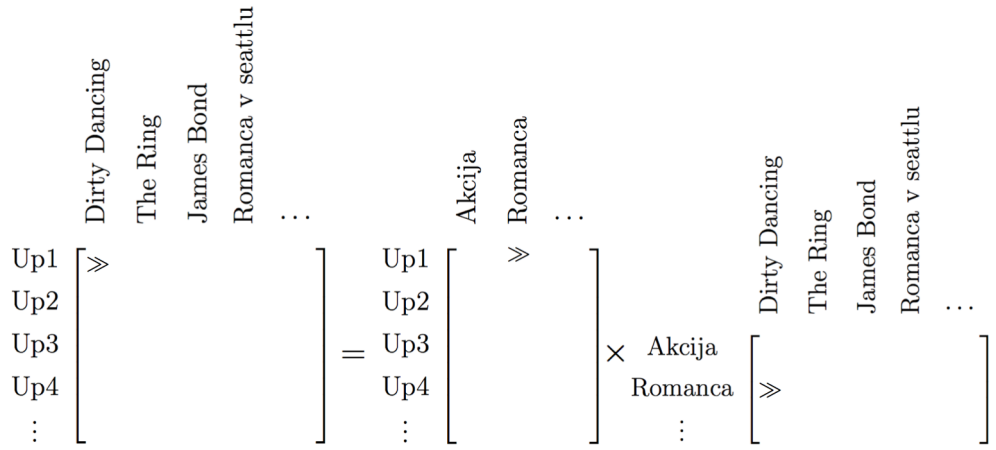
Najbolj razširjena je nenegativna matrična faktorizacija nad aritmetičnim polkolobarjem, pri kateri so vsi elementi matrik A, B in C nenegativna števila. Veliko se uporablja na področju bioinformatike, pri obdelavi zvočnih spektogramov ali mišične aktivnosti, kjer imajo podatki nenegativne vrednosti. Uporabljena pa je tudi v priporočilnih sistemih, ki denimo v spletnih trgovinah priporočajo izdelke uporabnikom glede na njihove preference. Eden danes najbolj uporabljenih pristopov grajenja takšnih sistemov je sodelovalno filtriranje [11]. Glavna ideja sodelovalnega filtriranja je poiskati

podobne uporabnike, ki so že ocenili podobne izdelke in uporabiti te informacije za priporočila drugim uporabnikom. Leta 2009, po končanem Netflix tekmovanju [14], je bilo dokazano, da lahko matrična faktorizacija pripomore k izboljšanju natančnosti priporočilnih sistemov.

Matrika A, ki je potrebna za izvedbo matrične faktorizacije, je sestavljena iz uporabnikov (v vrsticah) in izdelkov, ki so jih uporabniki ocenili (v stolpcih). Po faktorizaciji dobimo približne ocene, ki bi jih uporabniki najverjetneje dodelili še neocenjenim izdelkom in jim na podlagi teh priporočamo nove izdelke. Zanimivost novonastalih matrik B in C je, da medtem ko vrstice matrike B predstavljajo uporabnike, stolpci matrike C izdelke, stolpci matrike B in vrstice matrike C predstavljajo novo lastnost, preko katere se uporabniki povežejo z določenimi izdelki. V primeru, da so izdelki filmi, bi ta nova lastnost lahko bil tip filma, npr. romanca, grozljivka, triler,... Prvi stolpec matrike B oz. vrstica matrike C bi predstavljal romanco, drugi stolpec oz. vrstica grozljivko itd. Tako bosta v primeru, da je prvi uporabnik filmu Dirty Dancing dodelil visoko oceno, elementa $B_{1,2}$ in $C_{2,1}$ visokih vrednosti (slika 3.1). Po končani matrični faktorizaciji lahko iz končnih matrik B in C vidimo, kakšne so lastnosti posameznih uporabnikov in filmov.

Informativnost matrične faktorizacije je precej odvisna od izbire parametra k. Večje je število k, bolj je zmnožek matrik B in C podoben prvotni matriki A, vendar bo posledično težja interpretacija rezultatov. Prav tako ni dobro, da izberemo premajhen k. Takrat se bo novonastala matrika preveč razlikovala od A in rezultati ne bodo natančni. Z optimalno izbiro parametra k, je iz matrik B in C mogoče enostavno razbrati lastnosti k skupin. V primeru uporabnikov in filmov se tvorijo skupine glede na vrsto filma.

Drugi polkolobar, ki je z matrično faktorizacijo prinesel dobre rezultate, je max-krat polkolobar. Najbolj poznana algoritma matrične faktorizacije nad max-krat polkolobarjem sta Capricorn [13] in Cancer [12], ki zaradi operatorja max na mestu seštevanja delujeta po načinu "zmagovalec-vzame-vse". V primerjavi z bolj poznano nenegativno matrično faktorizacijo, ki deluje po načinu "deli-celote", matrični faktorji nad polkolobarjem $([0,1], \max, \times)$



Slika 3.1: Matrična faktorizacija matrike, katerih vrstice predstavljajo uporabnike, stolpci pa filme.

izpostavijo najbolj dominantne lastnosti in tako zgradijo ostrejši kontrast med tem, kar je in kar ni pomembno za določen faktor. Posledično jih tako lažje interpretiramo. Primer dobre uporabe algoritmov matrične faktorizacije max-krat polkolobarja je iskanje robov na slikah.

V nadaljevanju bomo podrobneje raziskali algoritem Cancer, ki smo ga implementirali in nadgradili v programu Matlab, da ga lahko izvajamo nad poljubnim polkolobarjem. Algoritem Cancer smo izbrali, saj za razliko od Capricorna, ki je bil ustvarjen za delovanje nad diskretnimi podatki, deluje nad realnimi števili. Prav tako z algoritmom Cancer v primerjavi z algoritmom Capricorn dobimo manjšo končno napako med vhodno matriko in produktom izhodnih matrik.

3.2 Algoritem Cancer

Cancer je algoritem matrične faktorizacije, ki se izvaja nad max-krat polkolobarjem, v katerem bo prevladovalo delo z matrikami. Z A_i bomo označevali i-to vrstico matrike A, z A^j pa njen j-ti stolpec. Matrika A brez i-tega stolpca bo zapisana kot A^{-i} oziroma brez j-te vrstice kot A_{-j}. Izraz blok bomo upo-

rabljali za matriko ranga 1. Nad polkolobarji ni ekvivalence med različnimi definicijami ranga matrik, kot jo poznamo nad aritmetičnim polkolobarjem. Tu se bomo osredotočili na tako imenovan produktni rang [4].

Definicija 3.1 *Rang matrike $A \in \mathbb{R}_+^{n\times m}$ je najmanjše naravno število k za katerega obstajata matriki $B \in \mathbb{R}_+^{n\times k}$ in $C \in \mathbb{R}_+^{k\times m}$, da velja $A = BC$. To je najmanjše število k, za katerega je lahko matrika A zapisana kot*

$$A = F_1 \oplus F_2 \oplus \cdots \oplus F_k,$$

kjer je F_i matrika ranga 1. Torej je $F_i \in \mathbb{R}_+^{n\times m}$ po definiciji oblike $F_i = bc$, kjer sta $b \in \mathbb{R}_+^{n\times 1}$ in $c \in \mathbb{R}_+^{1\times m}$.

Vhodni parametri algoritma so:

- matrika A,
- rang k,
- naravno število M, pove koliko ciklov bo naredil algoritem oziroma koliko krat bo obdelan vsak od k blokov,
- parameter $t \in \mathbb{N}$, ki predstavlja maksimalno stopnjo polinoma. Po vsaki iteraciji se stopnja polinoma poviša, ko pa doseže stopnjo enako t, se ponovno nastavi na privzeto vrednost 2,
- parameter $f = (0, 1)$, ki pove kolikšen delež bloka se razkrije pri vsaki ponovitvi.

Algoritem ima dva izhodna podatka in to sta matriki $B \in \mathbb{R}_+^{n\times k}$ in $C \in \mathbb{R}_+^{k\times m}$, katerih produkt je max-krat aproksimacija matrike A.

Želja algoritma Cancer je, da je $B \otimes C$ karseda blizu A, kjer so $A \in \mathbb{R}_+^{n\times m}$, $B \in \mathbb{R}_+^{n\times k}$ in $C \in \mathbb{R}_+^{k\times m}$ ter k pozitivno celo število. Bližino merimo s Frobeniusovo normo [19].

Definicija 3.2 *Frobeniusova norma je matrična norma $n \times m$ matrike A, definirana kot kvadratni koren vsote absolutnih kvadratov njenih elementov, računsko zapisana kot*

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}.$$

Na podlagi definicije 3.2 želimo minimizirati izraz

$$E(A, B, C) = \|A - B \otimes C\|_F^2 = \sum_{i,j}(A_{ij} - (B \otimes C)_{ij})^2. \tag{3.1}$$

Algoritem 1 predstavlja psevdokodo algoritma Cancer, katerega glavno delovanje je iterativno posodabljanje blokov s proceduro UpdateBlock. UpdateBlock spremeni enega izmed blokov, medtem ko ostali ostanejo nespremenjeni. Nato se trenutna dekompozicija matrike A primerja z do sedaj najboljšo in se v primeru izboljšave zamenja s trenutno. Zadnji korak je višanje stopnje aproksimacijskih polinomov. Nižja je stopnja, večja je možna širina za spreminjanje spremenljivk, višja je stopnja, boljša je aproksimacija polinoma.

3.2.1 Procedura UpdateBlock

Procedura UpdateBlock, katere psevdokoda je zapisana v algoritmu 2, vsako iteracijo algoritma Cancer posodablja blok $bc \in \mathbb{R}_+^{n\times m}$. Ta je sestavljen iz dveh vektorjev $b \in \mathbb{R}_+^{n\times 1}$ in $c \in \mathbb{R}_+^{1\times m}$. UpdateBlock izmenično posodablja le en element vektorja b in c s funkcijo AdjustOneElement. Ker je vsak blok $bc \in \mathbb{R}_+^{n\times m}$ sestavljen iz $n + m$ elementov, je maksimalno število elementov, ki jih lahko spremenimo, enako $\left\lfloor \frac{f(n+m)}{2} \right\rfloor$.

3.2.2 Procedura AdjustOneElement

Procedura AdjustOneElement (algoritem 3) vsakič posodobi en element stolpca b oziroma vrstice c. Za razlago predpostavimo, da medtem, ko je b fiksen, se

Algorithm 1 Psevdokoda algoritma Cancer

Input: $A \in \mathbb{R}_+^{n\times m}, k > 0, M > 0, t > 2, 0 < f < 1$

Output: $B^* \in \mathbb{R}_+^{n\times k}, C^* \in \mathbb{R}_+^{k\times m}$

function CANCER(A, k, M, t, f)
  $B = B^* = 0^{n\times k}, C = C^* = 0^{k\times m}$
  $bestError = E(A, B, C)$
  $deg = 2$
  for $count = 0$ **to** $k \times M - 1$ **do**
    $i = (count \bmod k) + 1$
    $N = B^{-i} \otimes C_{-i}$
    $[B^i, C_i] =$ UPDATEBLOCK(A, N, B^i, C_i, deg, f)
    if $E(A, B, C) < bestError$ **then**
      $B^* = B, C^* = C$
      $bestError = E(A, B, C)$
    end if
    if $count > k$ **and** $(count \bmod k) = 0$ **then**
      $deg = deg + 1$
    end if
    if $deg > t$ **then**
      $deg = 2$
    end if
  end for
  return B^*, C^*
end function

Algorithm 2 Psevdokoda procedure UpdateBlock

Input: $A \in \mathbb{R}_+^{n\times m}, N \in \mathbb{R}_+^{n\times m}, b \in \mathbb{R}_+^{n\times 1}, c \in \mathbb{R}_+^{1\times m}, deg \geq 2, 0 < f < 1$

Output: $b \in \mathbb{R}_+^{n\times 1}, c \in \mathbb{R}_+^{1\times m}$

1: **function** UpdateBlock(A, N, b, c, deg, f)
2: $niters = \left\lfloor \frac{f(n+m)}{2} \right\rfloor$
3: **for** $count = 1$ **to** $niters$ **do**
4: $c =$ AdjustOneElement(A, N, b, c, deg)
5: $b =$ AdjustOneElement(A^T, N^T, c^T, c^T, deg)^T
6: **end for**
7: **return** b, c
8: **end function**

vektor c spreminja. Ker se spremeni vrednost samo enega elementa, je potrebno preveriti kako sprememba vsakega od m elementov vektorja c vpliva na rešitev in tako izbrati tistega, ki prinese največjo izboljšavo. Vsak element c_l vpliva na napako po stolpcu l. Recimo, da posodabljamo blok z indeksom q. Naj bo N dekompozicijska matrika brez q-tega bloka, $N = B^{-q} \otimes C_{-q}$. Minimiziranje $E(A, B, C)$ v odvisnosti od c_l je enako minimizaciji

$$\gamma(A_l, N_l, c_{,l}) = \sum_{i=1}^{n} (A_{il} - \max\{N_{il}, b_i c_l\})^2. \tag{3.2}$$

Nato se namesto direktnega minimiziranja pokliče procedura PolyMin, ki je natančneje opisana v razdelku 3.2.3. V njej se izračuna aproksimacijski polinom, ki je ustrezen za minimizacijo. Ker nas zanima izboljšava, pridobljena s posodobitvijo trenutno izbranega elementa c, se izračuna razlika obeh minimizacij ter se shrani v vektor u. Ko se isti postopek izvede nad vsemi elementi vektorja c, se posodobi element, katerega indeks je enak indeksu največje vrednosti vektorja u. Tako se posodobi element, ki pripomore k največji izboljšavi.

Algorithm 3 Psevdokoda procedure AdjustOneElement

Input: $A \in \mathbb{R}_+^{n\times m}, N \in \mathbb{R}_+^{n\times m}, b \in \mathbb{R}_+^{n\times 1}, c \in \mathbb{R}_+^{1\times m}, deg \geq 2$

Output: $c \in \mathbb{R}_+^{1\times m}$

1: **function** UpdateBlock(A, N, b, c, deg)
2: **for** $j = 1$ **to** m **do**
3: $baseError = \sum_{i=1}^{n}(A_{ij} - \max\{N_{ij}, b_i c_j\})^2$
4: $[err, x_i] =$ PolyMin(A^j, N^j, b, deg)
5: $u_i = baseError - err$
6: **end for**
7: $i =$ indeks največje vrednosti vektorja u
8: $c_i = x_i$
9: **return** c
10: **end function**

3.2.3 Procedura PolyMin

Operirananje s funkcijo γ, definirano v 3.2, je zaradi njenih lastnosti, kot sta pomanjkanje konveksnosti in gladkosti zaradi operatorja maksimum, precej težavno. Zato jo namesto neposredne minimizacije aproksimiramo s polinomom g stopnje deg. Ko posodabljamo c_l, so vse ostale vrednosti funkcije fiksne, zato funkcijo γ zapišemo kot $\mu(x) = \gamma(A_l, N_l, b, x)$. Za aproksimacijo funkcije najprej izberemo $deg + 1$ točk na intervalu $(0, 1)$, $x_0, x_1, \ldots, x_{deg}$. Nato poiščemo polinom g, da bo

$$g(x_i) = \mu(x_i)$$

za i = 0, 1, ..., deg. To storimo z Lagrangevo interpolacijsko formulo in tako dobimo Lagrangev interpolacijski polinom [2], ki ga zapišemo kot

$$g(x) = \sum_{i=0}^{deg} \mu(x_i) l(x_i),$$

kjer je

$$l(x_i) = \prod_{\substack{j=0 \\ i \neq j}}^{deg} \frac{x - x_j}{x_i - x_j}, \quad i = 0, 1, \ldots, deg.$$

Poglejmo si primer izračuna Lagrangeovega interpolacijskega polinoma.

Primer 3.1 *Zapisati želimo interpolacijski polinom, ki se najbolj prileže točkam:* $(1, 3)$, $(2, -1)$ *ter* $(3, 1)$.

$$\begin{aligned} g(x) &= y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \\ &= 3\frac{(x-2)(x-3)}{(1-2)(1-3)} + (-1)\frac{(x-1)(x-3)}{(2-1)(2-3)} + \frac{(x-1)(x-2)}{(3-1)(3-2)} = \\ &= 3x^2 - 13x + 13. \end{aligned}$$

Ko imamo aproksimacijski polinom, poiščemo $x \in \mathbb{R}_+$, ki minimizira $g(x)$. Procedura vrača napako $g(x)$ ter optimalno vrednost x, pri kateri napako dosežemo.

Procedura PolyMin je implementirana v datoteki *cancer_poly_min.m*, v kateri je istoimenska funkcija z vhodnimi parametri $A\_j, N\_j, b, deg, semiring$ ter dvema izhodnima podatkoma err in x_i. V funkciji smo najprej definirali dva vektorja x in y, velikosti $deg+1$. V x smo shranili $deg+1$ naključnih števil med 0 in 1, v y pa vrednosti funkcije μ v istoležnih točkah vektorja x. Nato smo s funkcijo $lagrangepoly(x, y)$ [8] izračunali koeficiente interpolacijskega polinoma p, ki jih funkcija pridobi s pomočjo prej opisane Lagrangeove interpolacije. Naslednji korak je izračun prvega odvoda polinoma der s funkcijo $polyder(p)$ in iskanje vrednosti x_i, kjer odvod polinoma p enak 0, s funkcijo $roots(der)$. Zadnji korak je izračun vrednosti err, ki jo dobimo z izračunom $\mu(x\_i)$. Izhodna podatka procedure sta x_i in err.

3.2.4 Nadgradnja algoritma Cancer

Ker smo želeli, da algoritem Cancer deluje nad različnimi vrstami polkolobarjev, smo kot vhodni podatek algoritma 1 dodali parameter *semiring*.

Parameter *semiring* je celo število med 1 in 3, s katerim določimo polkolobar, nad katerim naj se algoritem izvaja. Tako smo na podlagi izbranega polkolobarja, ko se v algoritmu izvaja matrično množenje vedeli, katere operacije moramo za to uporabiti. Polkolobarji, nad katerimi smo izvajali Cancer, so max-krat, max-plus ter aritmetični polkolobar.

Poglavje 4

Manipulacija izrazno-dokumentnih matrik z algoritmom Cancer

Na osnovi članka [16] smo se odločili uporabili algoritem Cancer nad izrazno-dokumentnimi matrikami in primerjali rezultate z nenegativno matrično faktorizacijo. V tem poglavju bomo najprej pregledali teorijo izrazno-dokumentnih matrik, opisali podatke, iz katerih smo generirali izrazno-dokumentne matrike in predstavili rezultate, pridobljene z izvajanjem matrične faktorizacije nad temi matrikami.

4.1 Izrazno-dokumentne matrike

Izrazno-dokumentne matrike so posebna vrsta matrik, katerih vrstice predstavljajo izraze, stolpci pa dokumente. Element $a_{i,j}$ izrazno-dokumentne matrike A predstavlja, kolikokrat se je izraz i pojavil v j-tem dokumentu. Tako število pojavitev izraza v dokumentu se imenuje izrazna frekvenca. Za lažjo predstavitev smo v primeru 4.1 zbrali opise štirih filmov in zapisali izrazno-dokumentno matriko.

Primer 4.1 *Izbrali smo filme The Accountant, Moana, Allied in Billy Lynn's*

Long Halftime Walk. Prvi je triler, drugi anime, tretji in četrti pa sta drami. V tabeli je prikazanih nekaj najbolj pogostih in zanimivih izrazov.

		Dokument			
		The Accountant	Moana	Allied	Billy Lynn's...
Izrazi	action	3	0	0	0
	drama	1	0	1	1
	princess	0	2	0	0
	war	0	0	1	2

Tabela 4.1: Tabela frekvenc izrazov v opisih fimov.

Podatkom iz tabele pripada izrazno-dokumentna matrika

$$A = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}.$$

Vhodna matrika algoritma Cancer je matrika, katere elementi so realna števila med 0 in 1. Zato smo morali izrazno-dokumentne matrike, katerih elementi so izrazne frekvence, pretvoriti v ustrezno obliko in to tako, da ima normirana matrika seštevke elementov vseh stolpcev enake 1. To smo naredili tako, da smo vsak stolpec delili z vsoto elementov istega stolpca. Kot primer smo normirali matriko A iz primera 4.1 in dobili matriko

$$A_{norm} = \begin{bmatrix} 0.75 & 0 & 0 & 0 \\ 0.25 & 0 & 0.5 & 0.33 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0.67 \end{bmatrix}.$$

Za boljše razumevanje poteka normiranja poglejmo normo prvega stolpca:

$$A^1_{norm} = \frac{A^1}{\sum A^1} = \frac{\begin{bmatrix}3\\1\\0\\0\end{bmatrix}}{3+1+0+0} = \frac{\begin{bmatrix}3\\1\\0\\0\end{bmatrix}}{4} = \begin{bmatrix}0.75\\0.25\\0\\0\end{bmatrix}.$$

4.2 Generiranje izrazno-dokumentne matrike

Preden smo lahko začeli s testiranjem algoritma Cancer na realnih podatkih, smo morali napisati kratek program za branje besedil iz datotek in generiranje izrazno-dokumentne matrike. V Javi smo napisali razred *TermToDocument*, ki glede na določeno število datotek generira izhodno datoteko *output.txt*, v kateri so v vsaki vrstici zapisani izraz in frekvence izraza v vsakem od dokumentov. V takšni obliki je mogoče datoteko enostavno prebrati v Matlabu in pretvoriti vsebino v matriko, ki bo po normiranju vhodni podatek algoritma Cancer. Primer izhodne datoteke je na sliki 4.1.

```
ability 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2
basic 1 0 1 1 1 0 2 1 0 2 0 1 2 1 0 0 0 0 2 0 1
clustering 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
computer 1 0 0 1 0 0 1 0 0 3 0 0 0 0 0 0 0 0 2 0 0
elements 0 0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0
integral 0 0 0 0 0 0 1 0 0 0 4 0 0 0 0 0 0 0 0 1 0
network 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 5 0 0
...
```

Slika 4.1: Primer vsebine izhodne datoteke *output.txt*.

Da bi bila matrika izrazov čimbolj opisna, smo med branjem iz datoteke izločili vse veznike kot so a, and, the itd. in besede, ki ne dajo besedilu dodatnega pomena kot so again, give itd. Vse takšne izraze smo zbrali v datoteki *conjuctionWords_eng.txt*, da jo je enostavno dopolnjevati. Ker so

besede kot so mathematics in mathematical ter method in methods istega pomena, smo se odločili, da jih združimo v en izraz. Torej, če se v besedilu pojavita besedi method in methods, ne bosta v matriki nastala dva izraza s frekvencama 1, ampak en izraz s frekvenco 2. Vse zbrane skupine podobnih besed smo zbrali v datoteki *similarWords.txt*.

4.3 Podatki

Kot podatke za generiranje izrazno-dokumentne matrike smo se odločili uporabiti opise predmetov univerzitetnega programa prve stopnje računalništva in matematike. Predmetnik je sestavljen iz predmetov Fakultete za računalništvo in informatiko (FRI) [21] ter predmetov, ki se izvajajo na Fakulteti za matematiko in fiziko (FMF) [20]. Zbrali smo 21 angleških opisov in jih shranili v tektovne datoteke:

1. Analiza 1 (FMF),
2. Diskretne strukture 1(FMF),
3. Osnove digitalni vezij (FRI),
4. Programiranje 1 (FRI),
5. Linearna algebra (FMF),
6. Verjetnostni račun in statistika (FMF),
7. Analiza 2 (FMF),
8. Diskretne strukture 2 (FMF),
9. Programiranje 2 (FRI),
10. Arhitektura računalniških sistemov (FRI),
11. Analiza 3 (FMF),

12. Kombinatorika (FMF),
13. Algoritmi in podatkovne strukture 1 (FRI),
14. Osnove podatkovnih baz (FRI),
15. Izbrana poglavja iz matematike (FMF),
16. Optimizacijske metode (FMF),
17. Principi programskih jezikov (FRI)
18. Algoritmi in podatkovne strukture 2 (FRI)
19. Računalniške komunikacije (FRI)
20. Numerične metode (FMF)
21. Osnove umetne inteligence (FRI)

Generirano izrazno-dokumentno matriko smo v Matlabu prebrali iz *output.txt* datoteke ter izločili vse vrstice, katerih seštevek elementov je manjši od 2. Tako smo dobili matriko izrazov, ki se v opisih predmetov pojavijo vsaj 3-krat. Dobljeno matriko smo normirali in jo uporabili kot vhodni podatek algoritma Cancer. Ostali vhodni parametri algoritma so $\boldsymbol{k} = 2, 3$, $\boldsymbol{M} = 30$, $\boldsymbol{t} = 16$, $\boldsymbol{f} = 0.1$.

4.4 Rezultati in meritve

Generirana izrazno-dokumentna matrika 21ih opisov predmetov je matrika velikosti 202×21, ki vsebuje 202 različnih izrazov iz 21ih datotek. Po izvedbi matrične faktorizacije nad takšno matriko, smo glede na izbran k dobili izhodni matriki B in C, velikosti $202 \times k$ in $k \times 202$. Vrstice matrike B predstavljajo izraze, stolpci matrike C pa datoteke opisov predmetov. Stolpci matrike B in vrstice matrike C predstavljajo k lastnosti, ki povezujejo izraze z datotekami. Če zmnožimo matriki B in C, dobimo matriko $\tilde{A}$, ki je

karseda podobna začetni normirani matriki A. Matrično faktorizacijo smo nad vhodno matriko izvedli večkrat in vsakič uporabili drugačno vrednost k. Zanimalo nas je, če je mogoče glede na k izbrati imena novonastalih skupin in kaj nam te skupine dejansko povedo. Rezultate algoritma Cancer pri $k = 2$ smo primerjali z rezultati nenegativne matrične faktorizaicije. Zanimale so nas posebnosti in skupne lastnosti izhodnih matrik obeh algoritmov in kdaj je najbolj primerno uporabiti nenegativno matrično faktorizacijo in kdaj algoritem Cancer. Ker smo algoritem Cancer implementirali tako, da ga je mogoče izvesti nad poljubnim polkolobarjem, smo ga z isto vhodno matriko izvedli nad različnimi polkolobarji in primerjali rezultate.

4.4.1 Primerjava rezultatov glede na parameter k

Po izvedbi matrične faktorizacije nad vhodno matriko in parametrom $k = 2$, sta izhodni matriki B in C velikosti 202×2 in 2×202. To pomeni, da smo dobili dve novi skupini, po katerih lahko grupiramo izraze in dokumente. Poglejmo si nekaj vrstic matrike B in nekaj stolpcev matrike C:

$$B = \begin{bmatrix} 0 & 0.97551 \\ 0.15188 & 0 \\ 0 & 0.33841 \\ 0.079252 & 0 \\ 0.3494 & 0 \\ 0 & 0.89883 \\ 0 & 0.36676 \\ 0.10693 & 0 \\ \vdots & \vdots \end{bmatrix} \begin{matrix} algebra \\ database \\ differential \\ java \\ language \\ mathematics \\ vector \\ object \\ \vdots \end{matrix}$$

$$C = \begin{matrix} \\ \end{matrix} \begin{matrix} A3 & VRS & P1 & PSJ & \cdots \end{matrix} \\ \begin{bmatrix} 0.0069637 & 0 & 0.076306 & 0.10193 & \cdots \\ 0.019058 & 0.03112 & 0 & 0.0010389 & \cdots \end{bmatrix}$$

A3 = Analiza 3,
VRS = Verjetnostni račun in statistika,
P1 = Programiranje 1,
PSJ = Principi programskih jezikov.

Ker smo algoritem Cancer izvedli nad max-krat polkolobarjem, nam elementi zmnožka matrik B in C povedo, katera od dveh novonastalih lastnosti najbolj vpliva na to, da se nek izraz pojavi v določenem dokumentu. Torej, če pomnožimo vrstico matrike B, ki predstavlja izraz algebra z vsakim stolpcem matrike C, dobimo normirane frekvence izraza algebra v opisih predmetov v odvisnosti od ostalih izrazov v besedilu. Vrednosti stolpcev matrike B smo razvrstili glede na velikost ter poiskali izraze, ki pripadajo določenim vrednostim. Razvrščene izraze vsakega stolpca smo zapisali v stolpec Cancer tabele 4.2 kot skupina 1 in skupina 2. Iz tabele vidimo, da so bolj matematični izrazi zbrani v stolpcu skupine 1. Izrazi, ki so bolj računalniške narave pa so zbrani v stolpcu skupine 2. Če podobno naredimo z matriko C, so opisi matematičnih predmetov v eni skupini, opisi računalniških predmetov pa v drugi. Skupino 1 in skupino 2 lahko posledično poimenujemo FRI in FMF.

S parametrom $k = 3$ smo po izvedbi matrične faktorizacije nad vhodno matriko A dobili izhodni matriki B in C, tokrat velikosti 202×3 ter 3×202. Podobno kot pri $k = 2$, smo stolpce matrike B razvrstili po velikosti in v tabeli 4.3 zapisali izraze najvišjih vrednosti v posameznih stolpcih. Sestava izrazov prve skupine je podobna sestavi izrazov prve skupine pri $k = 2$, zato smo skupino 1 poimenovali računalništvo. 2. skupina izrazov iz $k = 2$ se je tukaj razdelila na dve novi skupini, skupina 2 in skupina 3. Izrazi skupine 3 so prisotni v matematičnih predmetih, v kateri prevladuje operiranje s funkcijami, integrali in enačbami, zato smo 3. skupino poimenovali matematična analiza. 2. skupino pa smo zaradi veliko algebraičnih izrazov poimenovali algebra.

Izhodne matrike so v obeh primerih matrične faktorizacije po svoje informativne. Ko je parameter $k = 2$, dobimo dve skupini, FRI in FMF, v

Cancer (k=2)		NNMF (k=2)	
Skupina 1	Skupina 2	Skupina 1	Skupina 2
program	function	program	basic
basic	number	language	problem
problem	equation	algorithm	theory
algorithm	discrete	data	discrete
solve	theory	design	number
data	application	basic	solve
language	algebra	problem	equation
design	analysis	develop	application
structure	theorem	ability	linear
method	mathematics	solve	algebra
ability	combinatorics	logic	structure
computer	statistics	type	mathematics
systems	linear	circuit	function
logic	course	computer	analysis
develop	test	skill	combinatorics
skill	solve	principle	systems
example	order	example	theorem
application	continuous	correctness	optimization
optimization	basic	semantic	graph
computation	concepts	method	algorithm
circuit	principle	database	method
type	integral	abstract	course
principle	conditional	c	model
database	science	systems	continuous
concepts	variable	computation	computer

Tabela 4.2: Izrazi prvih matrik algoritma Cancer in NNMF, ki so glede na vrednost elementa v stolpcu razvrščeni od najvišjega do najmanjšega.

Cancer (k=3)		
Skupina 1	Skupina 2	Skupina 3
program	basic	problem
algorithm	structure	solve
language	theory	equation
data	circuit	number
design	discrete	theorem
ability	algebra	basic
develop	application	method
principle	logic	optimization
problem	linear	principle
solve	mathematics	systems
computer	model	function
type	graph	computation
example	concepts	linear
skill	analysis	combinatorial
logic	course	network
abstract	design	independence
method	function	order
correctness	database	integral
semantic	systems	theory
c	digital	recursion
computation	number	combinatorics
various	matrix	configuration
transfer	combinatorics	coefficient
application	skill	set
different	transfer	computer

Tabela 4.3: Izrazi matrike B algoritma Cancer, ki so glede na vrednost elementa v stolpcu razvrščeni od najvišjega do najmanjšega.

katere se grupirajo predmeti in izrazi. Ko je $k = 3$, dobimo 3 skupine. Skupina FMF se v tem primeru razdeli na dve novi skupini. Višja bo vrednost parametra k, več bo nastalih skupin, dokler vsak izraz ne bo v svoji skupini.

V namišljenem primeru iz naših podatkov, ko npr. vemo, da je predmetnik za program računalništva in matematike sestavljen iz predmetov dveh fakultet, ne vemo pa, kateri predmet se izvaja na kateri od teh dveh, je primerna vrednost k enaka 2. V primeru, da bi 21 predmetom dodali predmete Fakultete za elektrotehniko, bi bila bolj primerna izbira vrednosti parametra k enaka 3.

Delovanje algoritma Cancer glede na k smo preverili tudi z izračunom napak med zmnožkom izhodnih matrik in vhodno matriko s Frobeniusovo normo ter izračunom časovne kompleksnosti izvajanja. Napaka, ko je $k = 2$, je 0.88 in 0.83, ko je $k = 3$. Napaka je pri manjšem k višja od napake pri višji vrednosti k. Po drugi strani se časovna kompleksnost izvajanja algoritma z večanjem k veča. Algoritem Cancer se pri k enakim 2 izvaja povprečno 225s, pri k enakim 3 pa 332s.

4.4.2 Primerjava z nenegativno matrično faktorizacijo

Rezultate algoritma Cancer, ko je parameter $k = 2$ smo primerjali z rezultati nenegativne matrične faktorizacije. V Matlab funkcijo $nnmf$ smo za vhodno matriko podali normirano izrazno-dokumentno matriko, kot drugi parameter pa vrednost 2. Izhodni matriki sta enakih velikosti kot izhodni matriki algoritma Cancer. V tabeli 4.2 drugi stolpec predstavlja rezultate prve izhodne matrike nenegativne matrične faktorizacije. Podobno kot za algoritem Cancer, smo tudi tukaj stolpca matrike B sortirali po velikosti in izpisali pripadajoče izraze. Na prvi pogled so si vsebine istoležnih stolpcev precej podobne. Izrazi so razvrščeni v dve skupini, prva vsebuje izraze, ki so značilni za FRI, druga pa izraze, značilne za FMF. Zaradi velike podobnosti smo izračunali Frobeniusovo normo, da bi videli ali se algoritma razlikujeta v višini napake med zmnožkom izhodnih matrik in vhodno matriko. Napaka algoritma Cancer je 0.88, napaka nenegativne matrične faktorizacije pa 0.87.

Ker tudi tukaj ni večjih razlik, smo se odločili preučiti vsebino izhodnih matrik. Če med sabo primerjamo prvi in drugi izhodni matriki obeh algoritmov končno opazimo razlike. Izhodno matriko B algoritma Cancer bomo označili z B_{cancer}, izhodno matriko nenegativne matrične faktorizacije pa z B_{nnmf}.

Matrika B_{cancer} ima večino vrstic takšnih, da je eden izmed elementov 0, drugi pa število, večje od 0. Ker se algoritem Cancer izvaja nad max-krat polkolobarjem, se elementi zmnožka izhodnih matrik izračunajo tako, da se na vsako mesto zapiše maksimalna vrednost zmnožkov istoležnih elementov vrstic matrike B_{cancer} in stolpcev matrike C_{cancer}. V zmnožku je tako vrednost le ena izmed lastnosti. Vrstice matrike B_{cancer} predstavljajo skupino, v kateri izraz prevladuje in kako pogosto se glede na druge izraze pojavlja v besedilih prevladujoče skupine. Ker smo do tega spoznanja prišli z razmišljanjem o delovanju algoritma Cancer nad max-krat polkolobarjem, smo ga želeli tudi računsko preveriti. Predmete iz matrike C_{cancer} smo glede na višjo vrednost v stolpcu razvrstili v dve skupini. Normirano vhodno matriko smo glede na prej definirano skupino razdelili na dve matriki. Prva je vsebovala stolpce prve skupine, druga stolpce druge skupine, vrstice obeh matrik pa so predstavljali izrazi. Ker element v takšnih matrikah predstavlja količino pojavitev določenega izraza glede na ostale izraze v besedilu, vsota takšnih elementov po vrsticah pove količino pojavitev določenega izraza v besedilih izbrane skupine. Istoležne vsote obeh matrik smo med seboj primerjali in izbrali skupino, ki je imela višjo vsoto. Tako smo izbrali skupino, v kateri se določen izraz glede na besedila v skupini pojavi večkrat. Če pogledamo nek izraz iz B_{cancer} in izračunano izbrano skupino izraza vidimo, da je vrednost elementa vrstice B_{cancer} večja od 0 tam, kjer je skupina enaka izbrani skupini. Za lažjo predstavo je v tabeli 4.4 prikazanih nekaj vrstic matrike B_{cancer} ter izračunane izbrane skupine izraza.

Najvišji vrednosti dveh skupin matrike B_{cancer} tako predstavljata izraza, ki se največkrat pojavita v opisih predmetov (v odvisnosti od ostalih izrazov v besedilu) iz določene skupine. To je enostavno preverljivo, če po velikosti razvrstimo elemente dveh prej izračunanih vektorjev vrstičnih vsot. Najvišjih

Izraz	B_{cancer} (Skupina 1)	B_{cancer} (Skupina 2)	Izbrana skupina
c	0.10289	0	1
arithmetic	0	0.42212	2
constructors	0.046471	0	1
language	0.3494	0	1
integral	0	0.56167	2
theorem	0	0.91241	2
statistics	0	0.81541	2
tree	0.077613	0	1
limit	0	0.32888	2

Tabela 4.4: Vsebina matrike B_{cancer} in izbrana skupina, v katero določen izraz glede na vrednosti matrike spada.

pet vrednosti obeh vektorjev predstavlja pet najbolj uporabljenih izrazov obeh skupin. Prvih pet izrazov prve skupine so program, basic, algorithm, data in problem. Če pogledamo prvi stolpec matrike B_{cancer} opazimo, da je vseh pet izrazov med prvimi šestimi vrsticami.

Za razliko od matrike B_{cancer}, so elementi matrike B_{nnmf} precej bolj zapolnjeni s števili, večjimi od 0. Število 0 se pojavi le na mestih, ko se izraz ne pojavi v nobenem opisu določene skupine predmetov. Takšen primer je vrstica izraza integral, katere prvi element je enak 0, drugi pa 0.04. To pomeni, da se izraz integral nikoli ne pojavi v prvi skupini. Za potrditev smo iz C_{nnmf} filtrirali predmete prve in druge skupine ter iz vhodne matrike pogledali vrednosti vrstice izraza integral glede na obe skupini. Vrednosti elementov prve skupine so vsi 0, torej se izraz integral nikoli ne pojavi v opisih predmetov prve skupine. Vsota elementov druge skupine pa je večja od 0, torej se je izraz integral v 2. skupini pojavil vsaj enkrat. Podobno velja za stolpce matrike C_{nnmf}. Vrednost elementa v stolpcih posameznih opisov je 0, ko besedilo ne vsebuje niti enega izmed izrazov ene od skupin. Na primer predmet Programiranje 2 ne vsebuje niti enega izmed izrazov iz

druge skupine, ki smo jo poimenovali FMF.

Če nekako povežemo ugotovitve iz prejšnjih odstavkov, sta si algoritem Cancer in nenegativna matrična faktorizacija v pogledu tvorjenja dveh skupin precej podobni. Pri obeh nastaneta skupini izrazov/dokumentov, ki spadajo v skupino Fakultete za računalništvo in informatiko ter skupina Fakultete za matematiko in fiziko. Najbolj se razlikujeta v vsebini prve izhodne matrike, ki vsebuje izraze v odvisnosti od skupin FRI in FMF. Vrstica matrike B_{cancer} nam pove, v katero skupino spada izraz, če bi ga morali dati v le eno od skupin. Vrstica matrike B_{nnmf} pa predstavlja delež pojavitve v eni in drugi skupini.

4.4.3 Primerjava rezultatov različnih polkolobarjev

Algoritem Cancer smo implementirali tako, da ga lahko uporabimo nad poljubnim polkolobarjem. V prejšnjih podpoglavjih smo se osredotočili na max-krat polkolobar, saj smo z njim dobili najbolj zanimive rezultate za interpretacijo in primerjavo z nenegativno matrično faktorizacijo. Algoritem Cancer smo poleg max-krat polkolobarja uporabili tudi nad aritmetičnim in max-plus polkolobarjem. Zanimalo nas je, kakšne rezultate v primerjavi z nenegativno matrično faktorizacijo dobimo, če izvedemo algoritem nad aritmetičnim polkolobarjem in kakšni bodo rezultati max-plus polkolobarja.

Ko smo izvedli algoritem Cancer nad aritmetičnim polkolobarjem, smo dobili rezultate zelo podobne rezultatom nenegativne matrične faktorizacije. Izračunana napaka med produktom izhodnih matrik in vhodno matriko je pri algoritmu Cancer nad aritmetičnim polkolobarjem 0.87, kar je enako kot pri nenegativni matrični faktorizaciji iz podpoglavja 4.4.2. Če primerjamo izraze drugega stolpca (NNMF) tabele 4.2 z izrazi algoritma Cancer so si zelo podobni. Edina večja razlika je pri časovni kompleksnosti. Povprečna časovna kompleksnost algoritma Cancer nad aritmetičnim polkolobarjem je 252s, v primerjavi s funkcijo $nnmf$, katere kompleksnost je manj kot 1s. Ker je razlika tako velika, se uporaba algoritma Cancer nad aritmetičnim polkolobarjem definitivno ne splača. Prav tako smo ugotovili, da izvajanje algoritma

Cancer nad max-plus polkolobarjem ne prinaša zanimivih rezultatov, ki bi jih bilo mogoče interpretirati, zato se nam jih ni zdelo vredno prikazati.

Poglavje 5

Napovedovanje proizvodnje električne energije obnovljivih virov

V tem poglavju bomo predstavili rezultate napovedovanja proizvodnje električne energije obnovljivih virov s pomočjo implementiranega algoritma Cancer. Najprej bomo na kratko predstavili podjetje, od katerega smo pridobili podatke, na podlagi katerih smo izvedli napovedi. Nato bomo opisali podatke, ki smo jih pretvorili v vhodno matriko algoritma Cancer ter ostale vhodne parametre, s katerimi smo dobili najboljše napovedi. Sledil bo pregled napovedi sončnega vira obnovljive energije, ki smo jih pridobili z izvajanjem algoritma Cancer nad max-krat polkolobarjem. Te bomo primerjali z dejansko proizvodnjo električne energije sončnih elektrarn, napovedmi podjetja SODO [18] ter napovedmi, pridobljenimi z izvajanjem nenegativne matrične faktorizacije. Preučili bomo delovanje algoritma Cancer nad max-plus in aritmetičnim polkolobarjem ter primerjali rezultate vseh treh polkolobarjev. Za zaključek pa bomo naredili še kratek pregled rezultatov napovedi vetrnih elektrarn in malih hidroelektrarn.

5.1 SODO d.o.o

Družba SODO, Sistemski operater distribucijskega omrežja z električno energijo, izvaja gospodarsko javno službo distribucijskega operaterja električne energije na ozemlju Republike Slovenije, za katero zagotavlja zanesljivo, varno in učinkovito oskrbo z električno energijo [17].

Družba SODO izvaja vsakodnevno 24 urno napovedovanje proizvodnje električne energije elektrarn obnovljivih virov za celotno Slovenijo. Glede na lokacijo spadajo v eno izmed petih elektrodistribucijskih podjetij: Elektro Celje, Elektro Ljubljana, Elektro Maribor, Elektro Gorenjska in Elektro Primorska. Osredotočili smo se na tri vire obnovljive energije:

Sončna energija

Sončna energija je energija, ki od sonca prihaja v obliki sončnega sevanja. Sončno energijo je mogoče enostavno pretvoriti v električno energijo. Procesu pretvorbe sončne energije v električno rečemo fotovoltaika in ga je mogoče sprožiti s sončnimi celicami. V okolici Maribora je 226 sončnih elektrarn, katerih priključna moč se giblje med 7 in 999 kW.

Vodna energija

Vodna energija oz. hidroenergija je v Sloveniji najbolj razširjen vir obnovljive energije. Vodna energija se s pomočjo hidroelektrarn pretvarja v električno. V okolici Maribora je 23 malih hidroelektrarn, katerih priključna moč je med 22 in 3170 kW.

Vetrna energija

Vetrno energijo je mogoče s pomočjo mlinov na veter in močnega vetra pretvoriti v električno energijo. V Sloveniji vetrna energija zaradi pomanjkanja stalnih močnih vetrov ni preveč popularna. Na omrežje je priključenih le 10 vetrnih elektrarn, največja je v Sežani, ki spada v elektrodistribucijsko podjetje Elektro Primorska.

5.2 Podatki

Pridobljeni podatki vsebujejo informacije o vsakourni proizvodnji električne energije, določenega elektrodistribucijskega podjetja, za določen vir obnovljive energije. Dodane so še lastnosti, ki bi nam lahko pripomogle pri končnem napovedovanju, npr. padavine, temperatura, vlažnost itd. in jih bomo v nadaljevanju podrobneje spoznali [5].

Podatke smo dali v skupine glede na vir obnovljive energije in na elektrodistribucijsko podjetje ter jih shranili v svoje CSV datoteke. Tako smo napovedovali proizvodnjo električne energije za vsakega distributerja in vsak vir obnovljive energije posebej. V vsaki datoteki je 13 stolpcev:

DATUM

- Opis: Datum, za katerega beležimo proizvodnjo električne energije.
- Format zapisa: DD/MM/LL, npr. 01/01/17

URA

- Opis: Ura, za katero beležimo proizvodnjo električne energije.
- Format zapisa: Celo število med 0 in 23.

POVP TEMP

- Opis: Povprečna temperatura ozračja v stopinjah Celzijah na določen dan, ob določeni uri, za določeno področje (na primer Maribor).
- Format zapisa: Realno število.

PADAVINE

- Opis: Količina padavin v *mm* na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: Realno število.

SEVANJE

- Opis: Povprečno sevanje v $\frac{W}{m^2}$ na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: Realno število.

MAX SEVANJE

- Opis: Maksimalno sevanje v $\frac{W}{m^2}$ na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: Realno število.

VLAŽNOST

- Opis: Izmerjena vlažnost zraka v % na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: Realno število med 0 in 100.

HITROST VETRA

- Opis: Izmerjena hitrost vetra v $\frac{m}{s}$ na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: Realno število.

SMER VETRA

- Opis: Smer vetra v stopinjah na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: 0 stopinj oz. 360 stopinj = sever

OBLAČNOST

- Opis: Povprečna oblačnost v % na določen dan, ob določeni uri, za določeno področje.
- Format zapisa: Celo število med 0 in 100.

ASTRO DAN

- Opis: Obdobje astro dneva določimo tako, da pogledamo uro nastopa zore in uro nastopa mraka. Ob urah med zoro in mrakom astro dan označimo z 1, sicer z 0
- Format zapisa: 0 ali 1.

ASTRO URA

- Opis: Podana je relativno na poldne, kjer je astro ura 0. Negativna vrednost pomeni dopoldan, pozitivna pa popoldan.
- Format zapisa: Celo število med -12, ko je ura 0 in 11, ko je ura 23.

SUM EL. ENERGIJE

- Opis: Količina pridobljene električne energije v kWh na določen dan, ob določeni uri, določenega vira obnovljive energije, za določeno področje.
- Format zapisa: Realno število.

Za lažje razumevanje je na sliki 5.1 prikazan primer datoteke s podatki sončnega vira obnovljive energije vseh ur 1. avgusta 2017 elektrodistribucijskega podjetja Maribor. Vidimo, da so si podatki med seboj precej različni. Nekateri so diskretni podatki, drugi realna števila, ki so lahko tudi negativna itd. Ker algoritem Cancer sprejme realna števila med 0 in 1, smo vsak

```
datum,ura,povp temp,padavine,sevanje,max sevanje,vlaznost,hitrost vetra,smer vetra,oblacnost,astro dan,astro ura,sum el.energije
01/08/16,0,18.4,0,0,0,89.7,1.8,25,40,0,-13,0
01/08/16,1,18.2,0,0,0,94.3,1.7,40,40,0,-12,0
01/08/16,2,18,0,0,0,95.5,1.4,65,50,0,-11,0
01/08/16,3,17.8,0,0,0,96.4,0.9,105,60,0,-10,0
01/08/16,4,17.7,0,1,20.11035418,96.7,0.9,155,50,0,-9,0
01/08/16,5,18.3,0,43,70.38168535,95.3,1.5,165,40,1,-8,15.02
01/08/16,6,19.2,0,153,218.2709592,93.4,0.8,155,20,1,-7,693.02
01/08/16,7,20.4,0,273,378.1057702,89.7,1.2,155,30,1,-6,2262.84
01/08/16,8,21.3,0.1,369,529.9792022,85.7,1.1,65,40,1,-5,3445.44
01/08/16,9,22.2,0,453,660.1686729,80.8,1.6,45,60,1,-4,6206.2
01/08/16,10,22.6,0,410,759.118824,75.1,0.9,75,70,1,-3,6640.44
01/08/16,11,23,0.1,404,820.7080244,73.7,1.9,35,70,1,-2,6898.56
01/08/16,12,22.5,0.1,391,841.5888426,78.6,2,60,80,1,-1,6633.33
01/08/16,13,21.1,1,184,820.7080244,88,1.5,35,100,1,0,2864.69
01/08/16,14,20.1,1,99,759.118824,89.7,0.2,35,80,1,1,2056.68
01/08/16,15,20.6,0.2,163,660.1686729,89.2,2,15,80,1,2,1919.3
01/08/16,16,20.6,0,129,529.9792022,85.6,2.4,325,80,1,3,1728.22
01/08/16,17,19.2,0,81,378.1057702,82.7,4.8,325,70,1,4,915.81
01/08/16,18,17.9,0,31,218.2709592,78.1,5.1,315,70,1,5,768.73
01/08/16,19,17.8,0,2,70.38168535,72.8,4.4,325,80,1,6,424.03
01/08/16,20,18,0,0,20.11035418,68.9,4.1,325,70,1,7,10.03
01/08/16,21,18,0,0,0,67.5,3.5,330,80,1,8,0
01/08/16,22,18,0,0,0,65.7,3.1,330,50,0,9,0.02
01/08/16,23,17.7,0,0,0,64.6,2.7,340,40,0,10,0
```

Slika 5.1: Primer podatkov sončnega vira obnovljive energije.

element stolpca normirali tako, da smo izračunali:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)},$$

kjer je x vektor, ki ga želimo normirati, z_i pa i-ti normiran element. Ker zadnji stolpec matrike vedno predstavlja količino proizvedene električne energije, katerega zadnji element bomo želeli napovedati z matrično faktorizacijo, za minimum in maksimum normiranje zadnjega stolpca uporabimo lokalni minimum in maksimum vrednosti stolpca brez zadnjega elementa in vrednostima prištejemo oz. odštejemo tretjino. Tako dosežemo, da je napovedana proizvodnja električne energije za današnji dan lahko večja oz. manjša kot v prejšnjih dnevih. Več si lahko o napovednih matrikah preberemo v podpoglavju 5.4.

Na primer, normirana matrika matrike

$$\begin{matrix} P1 & P2 & P3 & \\ \left[\begin{matrix} 26.9 & 713 & 13065 \\ 27.8 & 635 & 13428 \\ 29.7 & 496 & 11777 \\ 22.7 & 453 & 11988 \\ 24.6 & 645 & ? \end{matrix}\right. & & & \begin{matrix} d-4 \\ d-3 \\ d-2 \\ d-1 \\ \left.\begin{matrix} d \end{matrix}\right] \end{matrix}$$

je enaka

$$\begin{matrix} P1 & P2 & P3 & \\ \left[\begin{matrix} 0.6 & 1 & 0.52 \\ 0.7 & 0.7 & 0.55 \\ 1 & 0.16 & 0.39 \end{matrix}\right. & & & \begin{matrix} d-4 \\ d-3 \\ d-2 \end{matrix} \\ \left[\begin{matrix} 0 & 0 & 0.41 \\ 0.27 & 0.74 & ? \end{matrix}\right. & & & \begin{matrix} d-1 \\ \left.\begin{matrix} d \end{matrix}\right] \end{matrix} \end{matrix} .$$

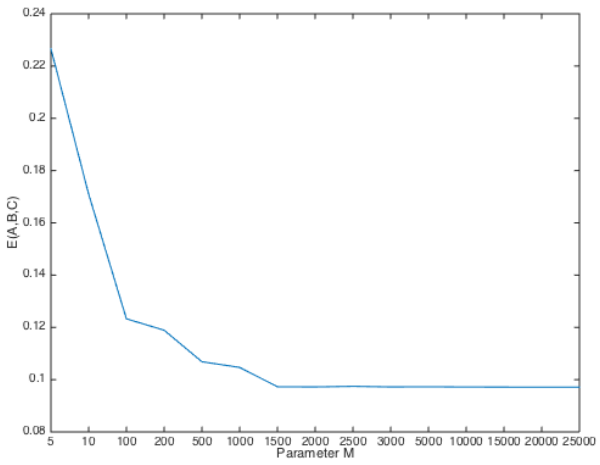
V takšni obliki je matrika primerna kot vhodni podatek algoritma Cancer. Vsak stolpec obeh matrik predstavlja vrsto podatka. V zgornjem primeru stolpec *P1* predstavlja povprečno temperaturo, stolpec *P2* predstavlja povprečno sevanje sonca in stolpec *P3* količino proizvedene energije. Vrstice obeh matrik pa predstavljajo zaporedne dneve ob določeni uri in to tako, da zadnja vrstica predstavlja današnji dan. Posledično vsaka vrstica predstavlja temperaturo, povprečno sevanje sonca in proizvodnjo energije za določen dan ob določeni uri. Zadnji element v matriki je neznan in ga bomo pridobili z matrično faktorizacijo.

Seveda vsi podatki niso uporabni za vsak vir obnovljive energije. Sklepamo lahko, da npr. na sončne elektrarne hitrost in smer vetra nimata velikega vpliva, za razliko od vetrnih elektrarn, kjer sta zelo pomembna podatka. Tako smo glede na vir obnovljive energije filtrirali stolpce za katere smo menili, da bi lahko vplivali na količino pridobljene energije.

5.3 Izbira vhodnih parametrov

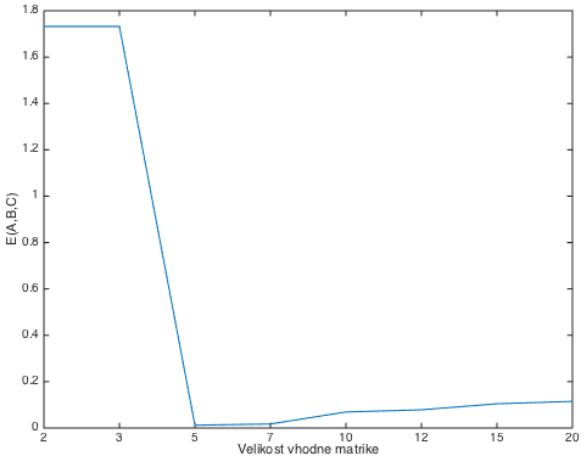
Želeli smo poiskati najboljšo kombinacijo vhodnih parametrov (k, M, t, f) ter velikost vhodne matrike, s katerimi bi bili rezultati glede na kompleksnost izvajanja algoritma kar se da najboljši.

Za začetek nas je zanimal vpliv števila iteracij M na delovanje algoritma, oziroma na njegovo konvergenco. Za M smo tako izbrali števila 5, 10, 100, 200, 500, 1000, 1500, 2000, 2500, 3000, 5000, 10000, 15000, 20000 in 25000. Opazovali smo vrednosti $E(A, B, C)$ v odvisnosti od števila iteracij. Iz grafa na sliki 5.2 lahko vidimo, da se vrednost $E(A, B, C)$ preneha izboljševati oz. algoritem začne konvergirati, ko je število iteracij večje od 1500.



Slika 5.2: Graf odvisnosti višine napake $E(A, B, C)$ od števila iteracij M.

Prav tako nas je zanimalo, kako velikost matrike oziroma število vrstic vpliva na končni rezultat. Natančneje, ali večje število dni pripomore k boljšim napovedim. Na grafu na sliki 5.3 vidimo, kako se spreminja vrednost napake med vhodno matriko in produktom izhodnih matrik. Algoritem smo pognali na vhodnih matrikah s številom vrstic 2, 3, 5, 7, 10, 15, 20. Vidimo, da je v primeru, ko je matrika sestavljena iz petih vrstic, napaka najmanjša.



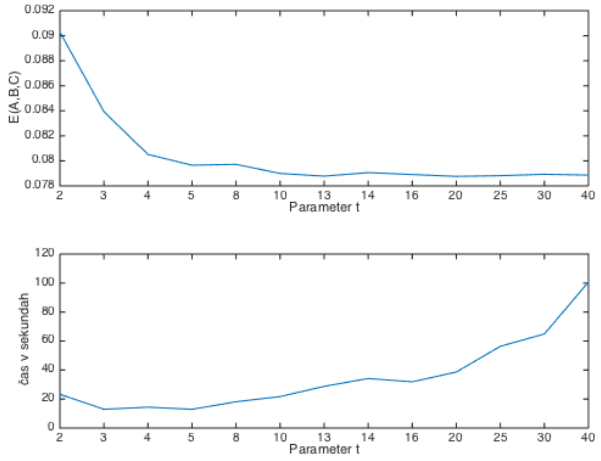
Slika 5.3: Graf odvisnosti višine napake $E(A, B, C)$ od velikosti matrike.

Za parameter f smo izbirali med vrednostmi 0.1, 0.3, 0.5, 0.6, 0.8, 1. V tabeli 5.1 sta prikazani časovna zahtevnost izvajanja algoritma in višina napake med vhodno matriko in produktom izhodnih matrik, glede vrednost parametra f. Napaka je najnižja, ko je parameter f enak 0.5 ali 1, vendar je pri $f = 1$ časovna kompleksnost algoritma dvakrat večja kot pri vrednosti $f = 0.5$. Zato smo za vrednost parametra f izbrali vrednost 0.5.

Parameter f	Časovna zahtevnost v sekundah	Napaka
0.1	0.34538	1.7321
0.3	19.9	1.7321
0.5	30.4	0.078785
0.6	30.204	0.080173
0.8	45.779	0.080363
1	63.187	0.078793

Tabela 5.1: Tabela odvisnosti časovne zahtevnosti algoritma Cacner in višine napake v odvisnosti od parametra f.

Parameter t predstavlja najvišjo stopnjo interpolacijskega polinoma, ki smo ga omenili v razdelku 3.2.3. Višja je stopnja polinoma, boljša in bolj natančna je aproksimacija funkcije, vendar se istočasno veča tudi časovna kompleksnost iskanja rešitve. Tudi tukaj smo na podlagi različnih vrednosti za parameter t preverili dolžino izvajanja algoritma in izračunali napako. Na sliki 5.4 smo prikazali dva grafa. Zgornji predstavlja odvisnost napake od izbranega parametra t, drugi pa odvisnost časovnega izvajanja algoritma v odvisnosti od parametra t. Želeli smo izbrati čimbolj optimalno vrednost t, ko je izvajanje algoritma čim krajše in napaka čim manjša. Na podlagi tega smo za najvišjo stopnjo interpolacijskega polinoma izbrali število 10.



Slika 5.4: Graf odvisnosti višine napake od parametra t in graf odvisnosti časovne zahtevnosti od parametra t.

Zadnji parameter, ki vpliva na delovanje algoritma Cancer je parameter k. Ta predstavlja število stolpcev matrike B ter število vrstic matrike C, oziroma število novih lastnosti, ki ustvarijo povezavo med lastnostmi vrstic in stolpcev začetne matrike. Ker smo želeli ugotoviti kako izbira parametra k vpliva na končno napoved, smo algoritem Cancer pognali z različnimi vrednostmi k: 2, 3, 4, 5, 10 in 12. Za dan 7.8.2017 smo za vsak k izračunali vsakourno napoved proizvodnje električne energije ter izračunali relativno

napako med vhodno matriko in zmnožkom matrik B in C. Napaka je najmanjša, ko je k enak 3 (16%) in za procent višja, ko je k enak 2. Za ostale vrednosti k je procent napake nad 30, kar je posledica prevelikega števila nastalih lastnosti, ki ne pripomorejo k izboljšavi izgradnje matrik B in C, da bi bil njun produkt bolj podoben vhodni matriki. Za nadaljnjo uporabo vrednost za $k = 2$ smo se odločili, saj je časovna zahtevnost izračuna matrične faktorizacije povprečno 15 sekund, rezultat pa le nekoliko slabši od rezultata pri vrednosti 3, za katero se algoritem izvaja povprečno 25 sekund.

Na tej točki smo definirali vse vrednosti vhodnih parametrov, da z matrično faktorizacijo nad definiranimi podatki dobimo kar se da najboljše rezultate. Če povzamemo, so parametri, ki smo jih pri testiranju uporabili kot vhodne podatke so $\boldsymbol{k} = 2$, $\boldsymbol{M} = 1500$, $\boldsymbol{t} = 10$, $\boldsymbol{f} = 0.5$.

5.4 Sončne elektrarne

Napovedi proizvodnje sončnih elektrarn za določen dan smo izvajali za vsako uro dneva med sončnim vzhodom in sončnim zahodom, ki se precej razlikuje glede letni čas. Vsakourna napoved je pridobljena z izvedbo algoritma matrične faktorizacije, tako smo algoritem Cancer za celodnevno napoved pognali tolikokrat, koliko ur je v astro dnevu (ur med sončnim vzhodom in zahodom). Vsakič smo dinamično generirali vhodno matriko, katere vrstice predstavljajo zaporedne dneve ob določeni uri. Torej, če želimo napovedati kakšna bo proizvodnja sončnih elektrarn danes ob 12h, bomo glede na to, koliko vrstično matriko želimo imeti, iz podatkov izbrali toliko preteklih zaporednih dni ob 12. uri. Tako smo zagotovili, da bo lega oz. višina sonca približno enaka in bo podatke mogoče lažje primerjati. Stolpci vhodne matrike predstavljajo vremenske lastnosti določenega dneva. Pri sončni energiji smo predvidevali, da na količino pridobljene energije najbolj vplivata temperatura zraka in sevanje sonca, zato sta to prva dva stolpca vhodne matrike. Tretji stolpec pa predstavlja dejansko količino proizvedene energije nekega dneva ob določeni temperaturi zraka in moči sevanja sonca.

Primer matrike za napovedovanje proizvodnje električne energije ob neki uri za dan d je

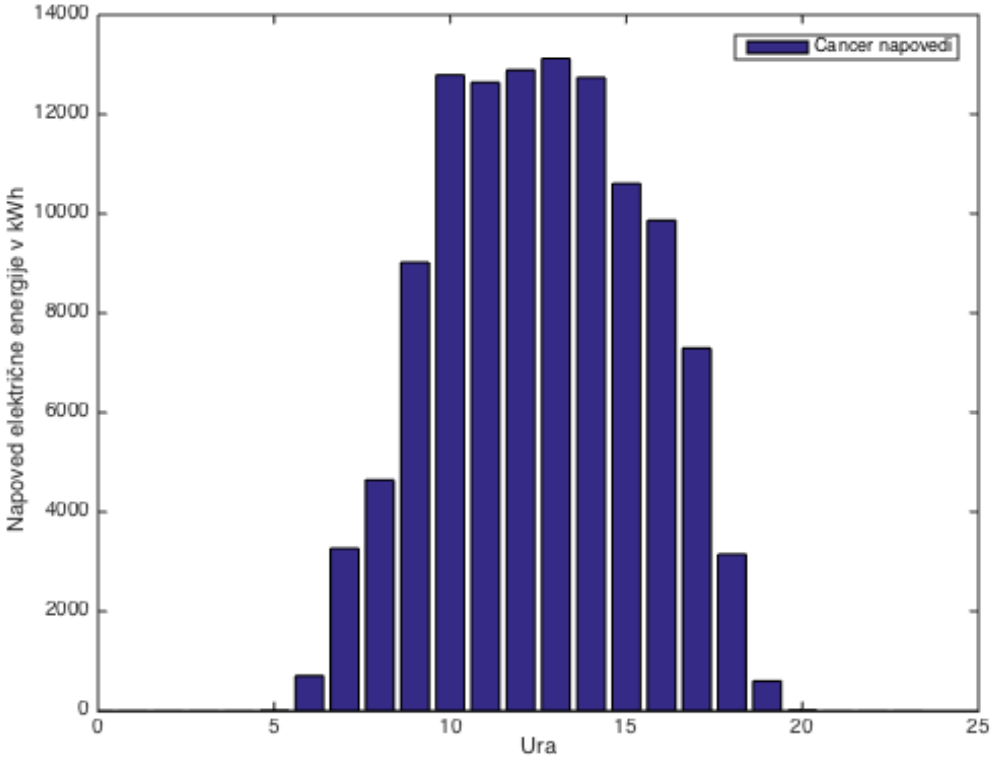
$$\begin{matrix} temp. & sevanje & el.energija & \\ \left[\begin{matrix} 26.9 & 713 & 13065 \\ 27.8 & 635 & 13428 \\ 29.7 & 496 & 11777 \\ 22.7 & 453 & 11988 \\ 24.6 & 645 & ? \end{matrix}\right. & & & \left.\begin{matrix} d-4 \\ d-3 \\ d-2 \\ d-1 \\ d \end{matrix}\right] \end{matrix} \cdot \tag{5.1}$$

Ker napovedujemo proizvodnjo v prihodnosti, nam podatki zadnje vrstice vhodne matrike manjkajo. Kljub temu je podatke o verjetni temperaturi zraka in sevanju sonca za nekaj dni naprej mogoče pridobiti iz javnih virov, denimo ARSO [1]. Tako nam manjka le zadnji podatek, na mesto katerega smo v matriki zapisali vprašaj in ga bomo z matrično faktorizacijo pridobili. Na to mesto pred začetkom izvajanja algoritma zapišemo število -1, da ga pri računanju napake $E(A, B, C)$ (3.1) ne upoštevamo, saj ne želimo da vpliva na izračun le te. To naredimo tako, da v primeru, ko je na zadnjem mestu vhodne matrike A število -1, to zamenjamo s številom 0. Prav tako na zadnje mesto produkta matrik B in C podamo število 0. Tako dosežemo, da je pri izračunu razlike med vhodno matriko A in produktom izhodnih matrik B in C, zadnji element matrike enak 0 in posledično ne vpliva na izračun napake.

Po izvedbi matrične faktorizacije nad normirano matriko matrike (5.1), dobimo matriki B in C katerih denormiran zmnožek je enak matriki

$$\begin{matrix} temp. & sevanje & el.energija & \\ \left[\begin{matrix} 26.9 & 698 & 13411 \\ 27.8 & 653 & 13058 \\ 29.7 & 496 & 11781 \\ 22.7 & 471 & 11605 \\ 24.6 & 646 & 13000 \end{matrix}\right. & & & \left.\begin{matrix} d-4 \\ d-3 \\ d-2 \\ d-1 \\ d \end{matrix}\right] \end{matrix} \cdot \tag{5.2}$$

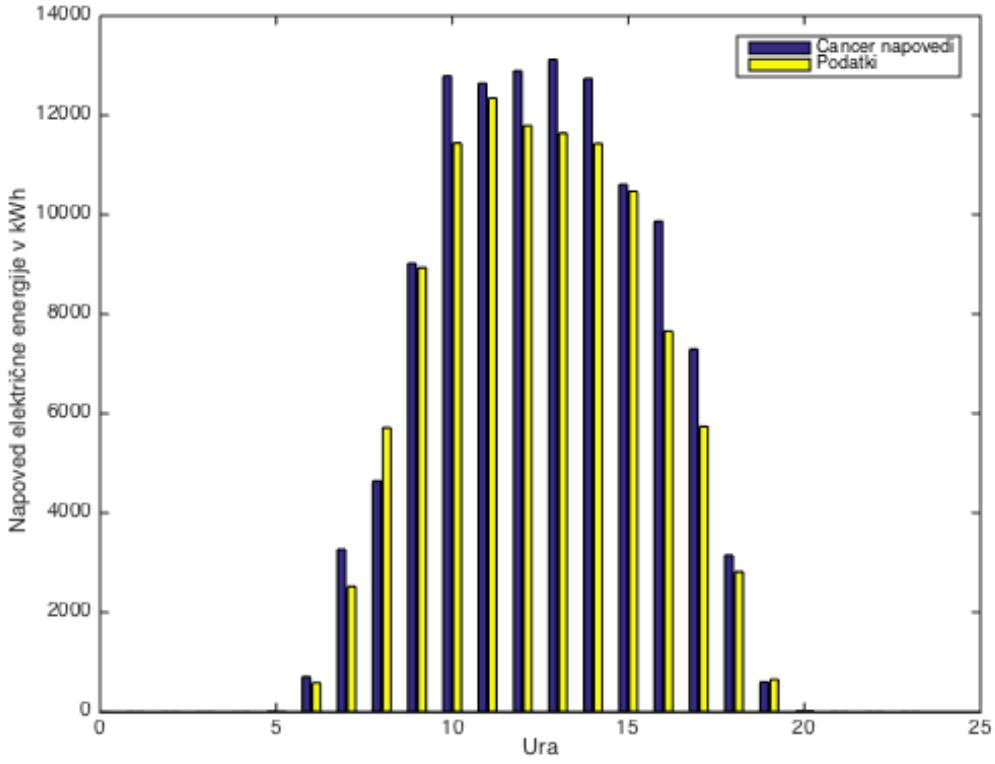
Kot napoved uporabimo zadnji element dobljene matrike (5.2), v tem primeru je ta 13000 kWh.



Slika 5.5: Stolpični diagram napovedi proizvedene energije v kWh za 7.8.2017.

Prvi dan, za katerega bomo prikazali napovedi proizvodnje sončnih elektrarn je 7.8.2017. Astro dan je poleti med 5. in 20. uro, kar pomeni, da smo matrično faktorizacijo izvedli 16-krat. Ker je izbran datum v preteklosti, imamo med podatki tudi dejanske vrednosti proizvodnje za tisti dan. Za potrebe izvajanja napovedi, smo zadnji element ignorirali ter ga obravnavali kot da ga med podatki ne bi bilo in ga kasneje uporabili za primerjavo z našimi napovedmi. Po končani izvedbi napovedi smo rezultate prikazali s stolpčnim diagramom, ki je na sliki 5.5. Os x predstavlja ure dneva, y os

pa našo napoved količine proizvedene energije v kWh. Iz napovedi vidimo, da se glede na lego sonca sorazmerno viša in manjša proizvodnja električne energije.



Slika 5.6: Stolpični diagram napovedi in dejanske proizvedene energije v kWh za 7.8.2017.

Ker smo napovedi želeli preveriti, smo na grafu na sliki 5.6 dodali rumeni stolpec z dejanskimi vrednostmi pridobljene sončne energije. Prav tako smo izračunali absolutno razliko med dejanskimi vrednostmi in napovedmi, iz česar je razvidno odstopanje napovedi proizvodnje. Ker proizvodnja sončne energije tekom dneva precej variira, nam absolutna napaka kot taka ne pove veliko, zato smo izračunali relativno napako za vsako uro dneva. To smo naredili tako, da smo absolutne napake delili z dejanskimi vrednostmi proi-

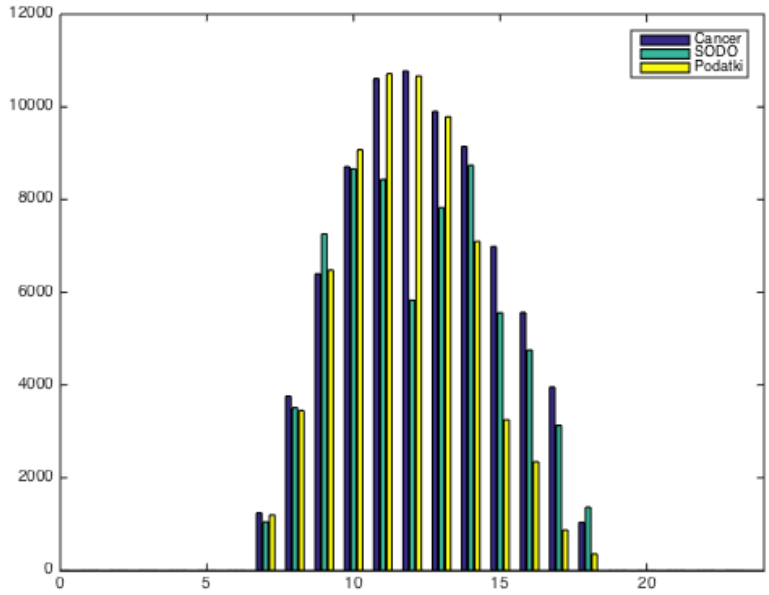
Ura	Dejanska proizvodnja	Napovedi	Absolutna napaka	Relativna napaka
5	6.12	10.673	4.553	74%
6	582.86	701.51	118.65	20%
7	2517.4	3266.8	749.4	30%
8	5712.3	4643.4	1068.9	19%
9	8930.9	9022.1	91.2	1%
10	11441	12790	1349	12%
11	12346	12643	297	2%
12	11789	12891	1102	9%
13	11641	13123	1482	13%
14	11431	12741	1310	11%
15	10470	10607	137	1%
16	7657.1	9865	2207.9	29%
17	5737.2	7292.8	1555.6	27%
18	2813.2	3146.7	333.5	12%
19	646.17	594.84	51.33	8%
20	13.91	12.576	1.334	10%

Tabela 5.2: Tabela rezultatov za 7.8.2017.

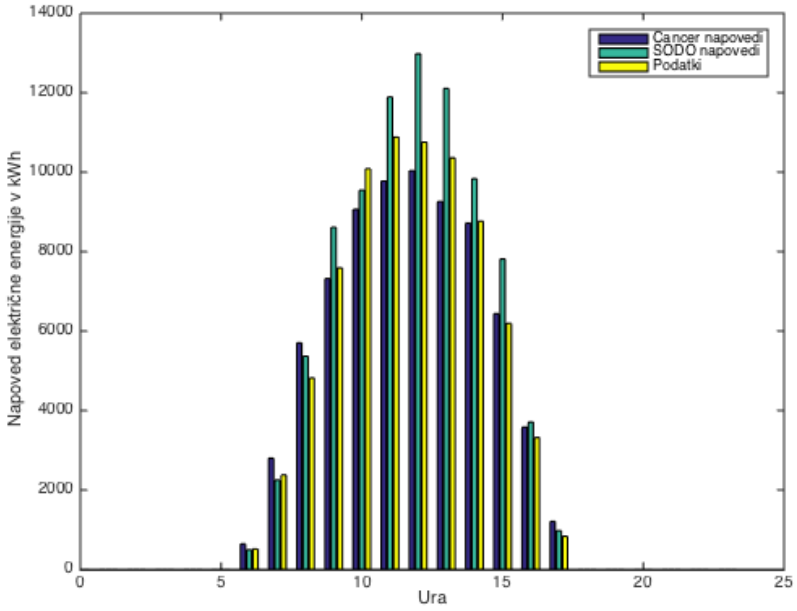
zvodnje in dobljene vrednosti množili s 100. Vsi rezultati so zbrani v tabeli 5.2. Na koncu smo izračunali še povprečno relativno napako in to tako, da smo sešteli napake v procentih ter vrednost delili s številom napovedi. Povprečna relativna napaka napovedi za 7.8.2017 je 17%.

Družba SODO tudi sama z metodo linearne regresije napoveduje vsakourno proizvodnjo električne energije. Zanimalo nas je, kako se njihove napovedi razlikujejo od naših in kako natančne dejansko so. Tako smo za 4.4.2017 izračunali napovedi z matrično faktorizacijo in jih primerjali z dejanskimi podatki ter napovedmi družbe SODO. Na grafu na sliki 5.7 smo dodali tretji stolpec zelene barve, ki predstavlja njihovo napoved. Ker iz grafa ni dobro razvidno katere napovedi so boljše, smo za obe izračunali ab-

solutno napako glede na dejansko proizvodnjo, nato pa še relativno napako. Rezultati so v tabeli 5.3. Prav tako smo za obe napovedi izračunali povprečno relativno napako, naša je bila 70%, SODOva pa 72%. Vidimo, da je v tem primeru napaka precej višja kot v prejšnjem, zato smo napovedi izvedli tudi za 25.3.2017, da bi videli, ali je pravilnost napovedi odvisna tudi od dneva. Izračunane napovedi 25.3.2017 so prikazane na grafu na sliki 5.8. Izračunana povprečna napaka je v tem primeru 19%. Tudi tukaj je SODOva napaka pri napovedovanju podobna naši in sicer 20%. Iz grafov vidimo, da kljub podobni relativni napaki, vsak algoritem dela napake za različne ure napovedovanja. Zakaj je temu tako, na žalost v okviru dela nismo ugotovili. Pojavi v naravi so zelo težko napovedljivi, saj občasno pride do nepredvidljivega obnašanja. Zato predvidevamo, da je napovedovanje nekaterih dni težje od drugih in so zato povprečne relativne napake za nekatere dneve višje od drugih. Razlog, da smo se odločili prikazati ravno ta dva dneva je to, da smo prikazali kako se možnost napovedljivosti iz dneva v dan spreminja.



Slika 5.7: Stolpični diagram naših in SODO napovedi ter dejanske proizvedene energije v kWh za 4.4.2017.



Slika 5.8: Stolpični diagram napovedi algoritma Cancer, SODO napovedi ter dejanskih vrednosti proizvedene energije v kWh za 25.3.2017.

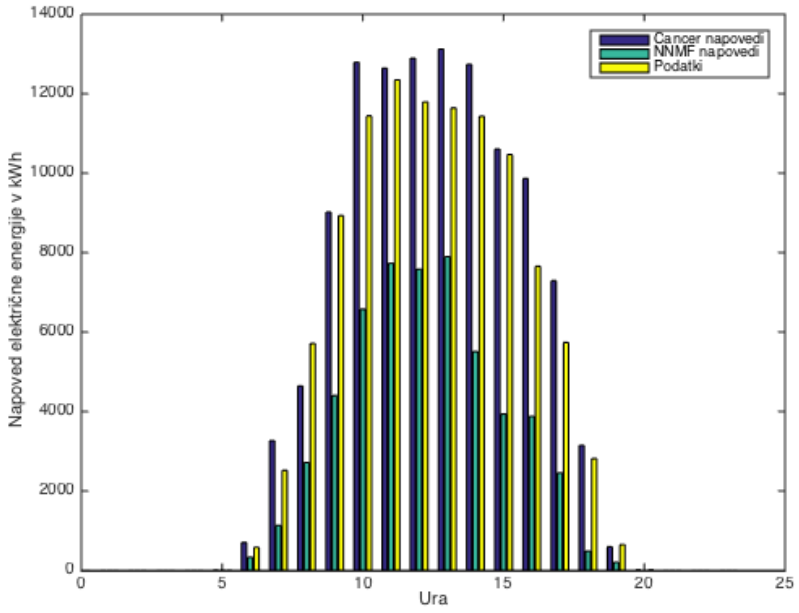
Ura	Dejanska proizvodnja	Naše napovedi	SODO napovedi	Absolutna (relativna) napaka naših napovedi	Absolutna (relativna) napaka SODO napovedi
7	1194.2	1243.1	1045.5	48.9 (41%)	148.7 (13%)
8	3448.1	3760.4	3509.9	312.3 (91%)	61.8 (18%)
9	6478.4	6397	7258.4	81.4 (13%)	780 (12%)
10	9076.3	8711.8	8662.7	364.5 (4%)	413.6 (46%)
11	10718	10612	8433.7	106 (1%)	2284.3 (21%)
12	10667	10774	5828.2	107 (10%)	4838.8 (45%)
13	9784.8	9903.8	7827.7	119 (12%)	1957.1 (20%)
14	7096.5	9146	8739.3	2049.5 (29%)	1642.8 (23%)
15	3249.2	6987.6	5556.5	3738.4 (115%)	2307.3 (71%)
16	2345.5	5564.6	4752.9	3219.1 (137%)	2407.4 (102%)
17	871.22	3955.9	3128.4	3084.7 (354%)	2257.2 (259%)
18	355.67	1037.1	1365.8	681.43 (191%)	1010.1 (284%)

Tabela 5.3: Tabela rezultatov naših in SODO napovedi za 4.4.2017.

Vsota vsakourne proizvodnje sončnih elektrarn je enaka količini proizvedene električne energije določenega dneva. Podobno je seštevek vsakournih napovedi enak predvideni celodnevni proizvodnji električne energije. Absolutna razlika med prejšnjima vsotama je absolutna napaka celodnevne napovedi v kWh. 25.3.2017 je bila dejanska celodnevna proizvodnja enaka 76496 kWh. Vsota naših napovedi je 74538 kWh, SODO pa 85576 kWh. Absolutna napaka naših napovedi je 1958 kWh (3.5%), SODO pa kar 9078 kWh (11.8%). Kljub temu, da sta povprečni relativni napaki proizvodnje zelo podobni, je celodnevna napaka napovedi podjetja SODO nekoliko višja od naše. Ker so SODO napovedi najslabše, ko je proizvodnja električne energije največja, je napaka celodnevne napovedi posledično višja. To se pri računanju vsako-

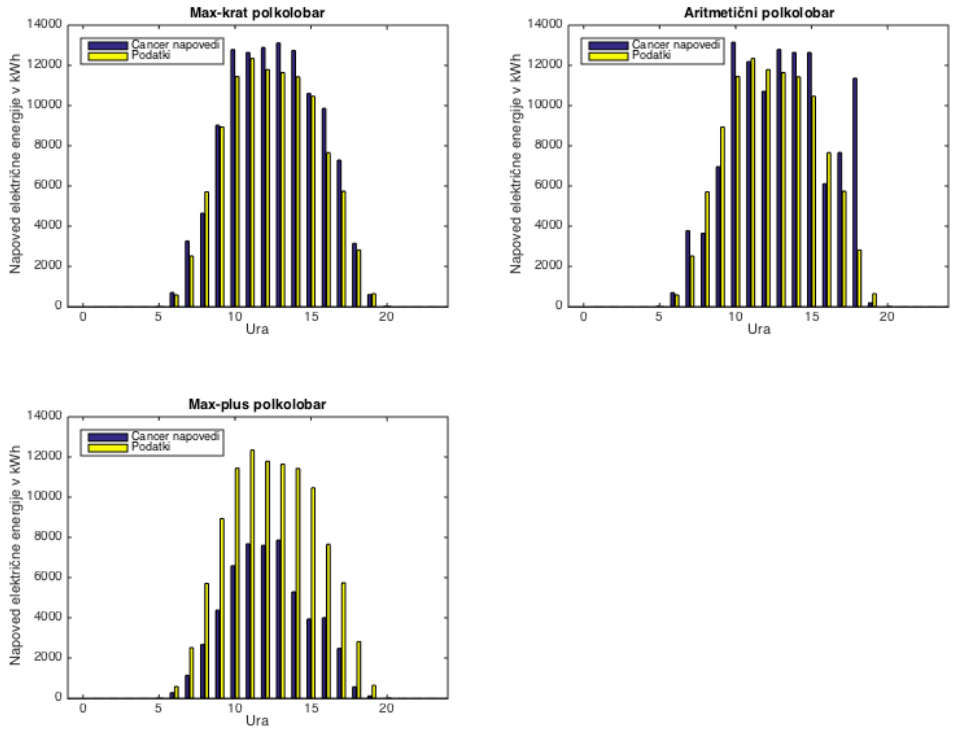
urne napake ne pozna, saj je izračun te odvisno le od ene ure oz. dejanske količine proizvedene energije v eni uri. Če izračunamo napovedi za 13. uro in izračunamo relativni napaki naših (10%) in SODO (16%) napovedi vidimo, da se napaki kljub temu, da na grafu izgleda, da je SODO napaka višja, razlikujeta le za 6%.

Ker je nenegativna matrična faktorizacija nad aritmetičnim polkolobarjem znana predvsem po tem, da daje dobre rezultate pri napovedovanju, smo jo želeli preizkusiti tudi na naših podatkih. Uporabili smo Matlab vgrajeno funkcijo $nnmf$ [9] in rezultate primerjali z našimi napovedmi ter dejanskimi vrednostmi proizvodnje električne energije. Funkcija $nnmf$ sprejme dva parametra. Prvi je matrika A, velikosti $n \times m$, za katero smo v našem primeru uporabili normirano matriko SODO podatkov ter rang k. Izhodna podatka sta tako kot pri algoritmu Cancer dve matriki velikosti $n \times k$ in $k \times m$. Denormiran zadnji element zmnožka izhodnih matrik je vrednost, ki jo obravnavamo kot napoved proizvodnje električne energije. Za 7.8.2017 smo že v tabeli 5.2 izračunali napake naših napovedi in iz teh dobili povprečno relativno napako 17%. Podobno smo naredili tudi z $nnmf$ rezultati in dobili precej višjo relativno napako 51%. Vidimo, da z algoritmom Cancer v primerjavi z nenegativno matrično faktorizacijo dobimo boljše in bolj natančne rezultate. Te rezultate smo grafično prikazali na grafu na sliki 5.9. Stolpci rumene barve so dejanske vrednosti proizvedene energije, modri stolpci predstavljajo naše napovedi, zeleni pa napovedi nenegativne matrične faktorizacije.



Slika 5.9: Stolpični diagram napovedi algoritma Cancer, napovedi $nnmf$ in dejanskih vrednosti proizvedene energije v kWh za 7.8.2017.

Za konec si poglejmo še primerjave delovanja implementiranega algoritma Cancer nad različnimi polkolobarji. Na sliki 5.10 so prikazani trije grafi. Na vsakem izmed grafov sta dva stolpca, rumen predstavlja dejansko količino pridobljene energije za 7.8.2017, modri pa napovedi različnih polkolobarjev. Na zgornjem levem grafu je napoved izračunana nad max-krat polkolobarjem, ki smo ga obravnavali na začetku tega podpoglavja. Zgornji desni graf predstavlja napovedi, ki so nastale pri uporabi aritmetičnega polkolobarja, spodnji levi graf pa napovedi, pridobljene ob uporabi max-plus polkolobarja. Podobno kot v prejšnjih primerih smo tudi tukaj izračunali povprečne relativne napake vsake od napovedi. Pri uporabi max-krat polkolobarja je napaka 17%, aritmetičnega 52% in max-plus 54%. Najboljše napovedi tudi tukaj dobimo z uporabo max-krat polkolobarja.



Slika 5.10: Grafi napovedi algoritma Cancer za 7.8.2017 nad različnimi polkolobarji.

5.5 Vetrne elektrarne in hidroelektrarne

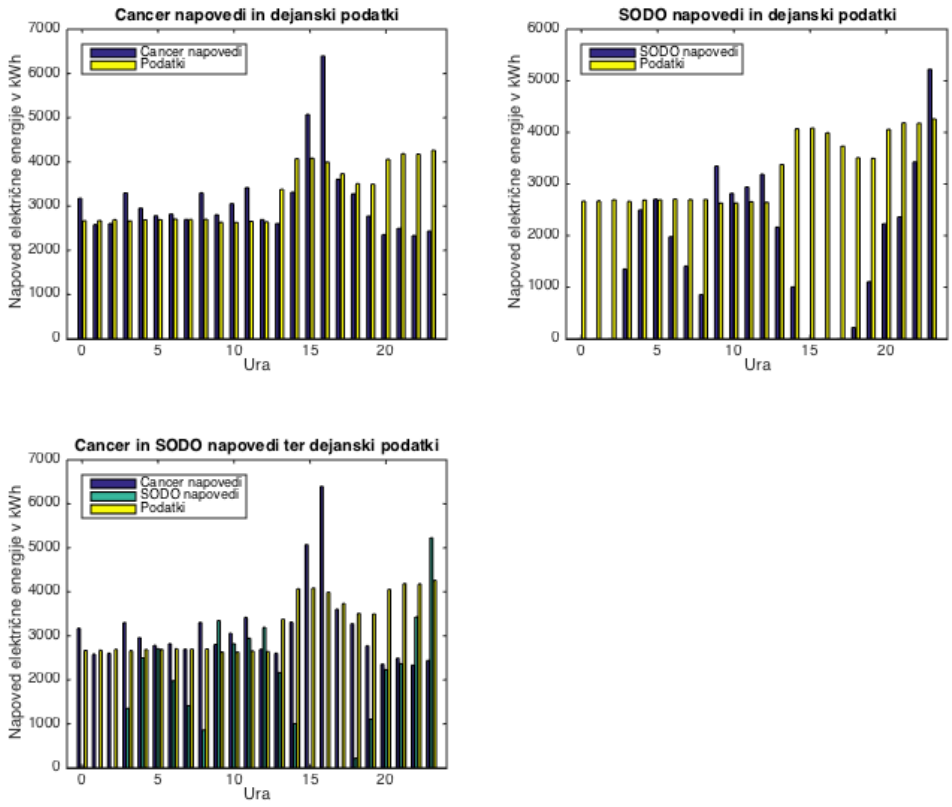
Algoritem smo preizkusili na podatkih sončnih elektrarn, hidroelektrarn in vetrnih elektrarn. V prejšnjem podpoglavju smo se osredotočili na sončne elektrarne, kjer so napovedi odvisne od moči in lege sonca, ki je v določenem letnem času na isti poziciji, moč se pa vsakodnevno glede na lego veča in manjša. Zato je največja proizvodnja sončnih celic opoldne, ko je sonce v najvišji legi. Za razliko od sončnih elektrarn, napoved proizvodnje vetrnih in hidroelektrarn ni tako enostavna.

Proizvodnja malih hidroelektrarn je odvisna od količine vode, ki teče po potokih, kjer so elektrarne postavljene. Med SODO podatki smo dobili informacije o vsakourni količini padavin, ki pade na nekem območju. To se nam je zdel glavni faktor, ki bi lahko vplival na količino proizvedene energije in smo ga zato uporabili pri napovedovanju. Pri napovedovanju sončnih elektrarn, smo vrstice matrike za matrično faktorizacijo sestavili iz zaporednih n dni ob določeni uri, tukaj pa smo se odločili uporabiti n zaporednih ur. Vhodna matrika za napovedovanje količine proizvedene energije malih hidroelektrarn za uro t je enaka

$$\begin{matrix} padavine & el.energija & \\ \left[\begin{array}{cc} 1.4 & 225.25 \\ 1.3 & 225.35 \\ 11.1 & 225.35 \\ 3.1 & 225.4 \\ 3 & ? \end{array}\right. & & \begin{array}{c} t-4 \\ t-3 \\ t-2 \\ t-1 \\ t \end{array} \end{matrix} .$$

Za 25.6.2017 smo z matrično faktorizacijo izračunali vsakourne napovedi proizvedene energije malih hidroelektrarn v območju Maribora. Med podatki so prav tako SODO napovedi, pridobljene z linearno regresijo. Na sliki 5.11 so prikazani trije grafi rezultatov. Na zgornjem levem so z modrim stolpcem prikazani rezultati naših napovedi, z rumenim pa dejanske vrednosti pridobljene energije za ta dan. Vidimo, da napovedi ob določenih urah precej odstopajo od dejanskih vrednosti. Informacije o padavinah so omejene na

večje območje, zato ni zagotovljeno, da je nevihtni oblak ravno nad potokom, na katerem je postavljena hidroelektrarna. Količina padavin v takšnem primeru ni primerljiva z dejansko količino padavin, ki so padle na predelu, kjer stoji hidroelektrarna, zato je napovedovanje oteženo. Na zgornjem desnem grafu modri stolpci predstavljajo napovedi podjetja SODO, rumeni pa dejanske vrednosti. Že iz grafov vidimo, da so v primerjavi z našimi napovedmi, SODO napovedi precej slabše. Izračunali smo tudi relativno napako vsake od napovedi in tako še potrdili grafične ugotovitve. Relativna napaka naših napovedi je 20%, njihova pa 53%. Na tretjem grafu smo združili zgornja dva, za lažjo primerjavo obeh napovedi. Modri stolpci so v tem primeru naše napovedi, zeleni SODO napovedi, rumeni pa dejanske vrednosti proizvodnje.



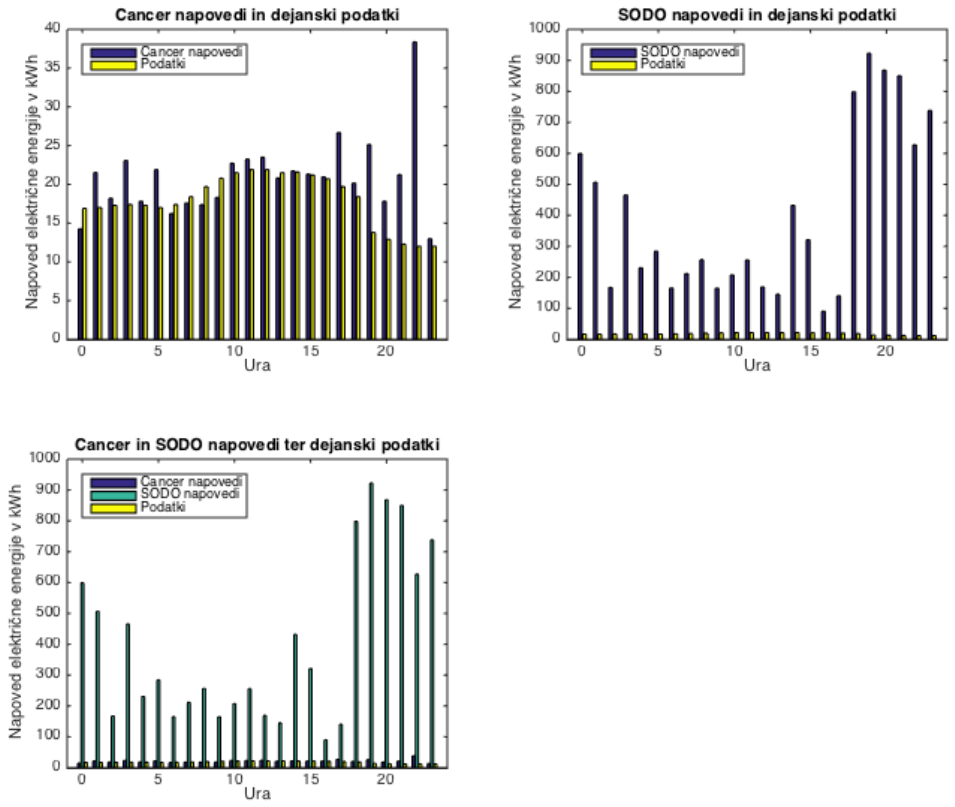
Slika 5.11: Grafi napovedi hidroelektrarn za 25.6.2017.

Matrika za napovedovanje proizvodnje vetrnih elektrarn je sestavljena iz treh stolpcev. Prvi predstavlja hitrost vetra, drugi smer vetra, tretji pa proizvedeno količino električne energije ob določeni uri. Tako kot pri hidroelektrarnah, tudi tukaj vrstice predstavljajo zaporedne ure in ne dneve ob istem času, kot pri matrikah sončnih elektrarn. Primer vhodne matrike za

napovedovanje proizvodnje elektrike vetrnih elektrarn za uro t je

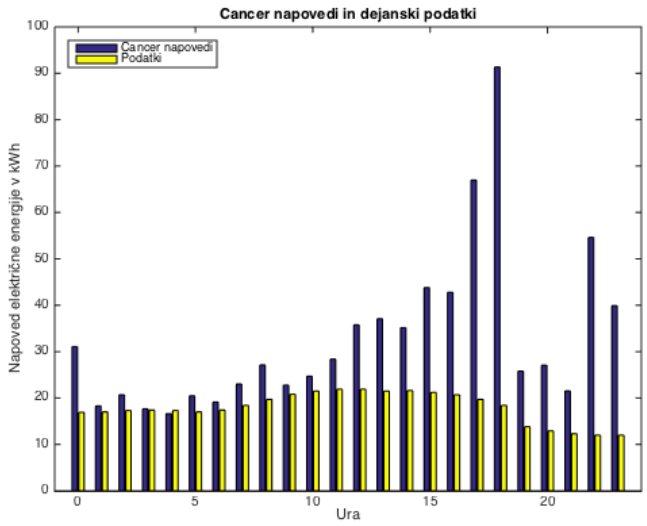
$$\begin{array}{ccc} hitrost\,vetra & smer\,vetra & el.energija \\ \left[\begin{array}{ccc} 3.6 & 115 & 21.5 \\ 3.9 & 120 & 21.9 \\ 4.4 & 125 & 21.9 \\ 4.9 & 125 & 21.5 \\ 4.6 & 120 & ? \end{array}\right] & \cdot & \begin{array}{c} t-4 \\ t-3 \\ t-2 \\ t-1 \\ t \end{array} \end{array}$$

Za napovedovanje smo iz podatkov izbrali enega najbolj vetrovnih dni, 7.6.2017, ter izvedli matrično faktorizacijo za vsako uro tega dneva. Med podatki smo poiskali informacije o napovedih podjetja SODO in jih podobno kot v prejšnjih primerih primerjali med sabo. Na sliki 5.12 so trije grafi. Prvi, zgornji levi, predstavlja rezultate naših napovedi v primerjavi z dejanskimi podatki. Drugi, zgornji desni, prikazuje primerjavo dejanskih podatkov in SODO napovedi. Tretji, spodnji levi, združuje prva dva grafa. Modri stolpci predstavljajo naše napovedi, rumeni dejansko proizvodnjo elektrike, zeleni pa SODO napovedi. Relativna napaka naših napovedi je 26%, napaka SODO napovedi pa kar 2134%. Zaradi hitre spremenljivosti hitrosti vetra je napovedovanje proizvodnje vetrnih elektrarn precej težko. Vidimo pa, da so kljub temu naše napovedi dobre v primerjavi z napovedmi podjetja SODO.



Slika 5.12: Grafi napovedi vetrnih elektrarn za 7.6.2017 glede na podatke zaporednih ur.

Napovedovanje proizvodnje električne energije s pomočjo matrične faktorizacije se je izkazalo za uporabno metodologijo. V primerih napovedovanja sončne energije, so naše napovedi približno enako dobre kot napovedi podjetja SODO. Najbolje se obnesejo pri napovedih vetrnih elektrarn, kjer so napovedi podjetja SODO precej slabše od naših. Seveda je velik razlog za to, uporaba podatkov zaporednih ur, kar je v praksi nemogoče, saj podatkov le za nekaj ur nazaj še nimamo. S takšnim izvajanjem smo želeli preveriti, kako dobro naša metodologija deluje, če napovedujemo na podlagi podatkov, za katere vemo, da so najbolj primerni za izračunavanje proizvodnje.



Slika 5.13: Graf napovedi vetrnih elektrarn za 7.6.2017 glede na podatke zaporednih dni.

Za primerjavo smo za isti dan, 7.6.2017, izvedli matrično faktorizacijo nad matriko podatkov vetrnih elektrarn, katere vrstice so tako kot v matrikah v razdelku 5.4, sestavljene iz zaporednih dni ob istih urah. Na sliki 5.13 je graf, na katerem so prikazane napovedi v primerjavi z dejanskimi vrednostmi proizvodnje. Vidimo, da so v primerjavi z napovedmi na prvem grafu na sliki 5.12, kot smo predvidevali, napovedi manj natančne. Izračunana relativna napaka je 90%, kar je v primerjavi z napako napovedi SODO, ki je 2134%, še vedno veliko manjša.

Poglavje 6

Sklepne ugotovitve

V magistrskem delu smo se osredotočili na raziskovanje polkolobarjev. Pogledali smo lastnosti, vrste in možnosti uporabe te zanimive strukture. Nato smo z algoritmom Cancer izvedli matrično faktorizacijo nad dvema tipoma podatkov. Glavni prispevki so tako implementiran algoritem in rezultati, ki smo jih z izvajanjem tega pridobili. Posebnost implementiranega algoritma, ki je bil ustvarjen za delo nad max-krat polkolobarjem je, da smo ga nadgradili tako, da deluje nad poljublnim polkolobarjem. Posledično smo lahko rezultate izvajanja algoritma nad podatki primerjali in izbrali tisti polkolobar, nad katerim je algoritem proizvedel najboljše rezultate.

Rezultati izvajanja algoritma nad izrazno-dokumentnimi matrikami so prinesli zanimive ugotovitve. Izrazno-dokumentne matrike smo zgradili iz opisov predmetov na Interdisciplinarnem študiju računalništva in matematike. Glede na predmetnik programa, ki je sestavljen iz predmetov dveh fakultet, smo z matrično faktorizacijo, brez da bi vedeli, na kateri fakulteti se posamezen predmet izvaja, dobili dve skupini predmetov glede na to, na kateri fakulteti naj bi se izvajal. Rezultate smo primerjali z nenegativno matrično faktorizacijo. V prihodnosti bi bilo zanimivo podobno narediti z moduli Fakultete za računalništvo in informatiko ali drugimi programi Univerze v Ljubljani.

Z matrično faktorizacijo smo prav tako poskušali napovedovati proizvo-

dnjo električne energije obnovljivih virov. Podjetje SODO, od katerih smo pridobili podatke, že samo s pomočjo linearne regresije izvaja izračune za napoved proizvodnje. Tako smo imeli možnost naše napovedi primerjati z njihovimi napovedmi in dejanskimi vrednostmi. V primeru napovedovanja proizvodnje sončnih elektrarn je napaka naših in njihovih napovedi približno enaka. V primeru napovedi malih hidroelektrarn in vetrnih elektrarn, pa so naše napovedi boljše. Podjetje SODO bi tako lahko našo metodologijo v primeru napovedovanja proizvodnje malih hidroelektrarn in vetrnih elektrarn uporabljalo kot glavni vir napovedovanja, za napovedovanje sončne energije pa kot primerjavo s svojimi napovedmi. Dobra smer nadaljnjega raziskovanja je iskanje razlogov, zakaj je napovedovanje nekaterih dni lažje od drugih ter za katere dneve je boljša napoved z matrično faktorizacijo nad max-krat polkolobarjem, za katere z nenegativno matrično faktorizacijo in za katere z linearno regresijo. Prav tako bi bilo dobro poiskati še kakšne, mogoče ne podnebne podatke, ki bi lahko vplivali na proizvodnjo električne energije. Algoritem bi lahko izvedli na podatkih vseh elektrodistribucijskih podjetjih in ostalih obnovljivih virih ter napovedi med seboj primerjali.

Literatura

[1] Agencija Republike Slovenije za okolje. http://www.arso.gov.si. [Dostopano: 18.9.2017].

[2] B. Archer and E. W. Weisstein. "Lagrange Interpolating Polynomial." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html. [Dostopano 6.8.2017].

[3] R. B. Bapat. *Graphs and matrices*, volume 27. Springer, 2010.

[4] L. B. Beasley and A. E. Guterman. Rank inequalities over semirings. *J. Korean Math. Soc*, 42(2):223–241, 2005.

[5] D. Bokal, S. Šmigoc, Ž. Povalej, M. Delčnjak, A. Veber-Horvat, SODO (Maribor)., and XLAB (Ljubljana). *Napovedovanje potrebne energije za pokrivanje izgub v distribucijskem omrežju*. 2015.

[6] R. A. Brualdi and D. Cvetković. *A combinatorial approach to matrix theory and its applications*. CRC press, 2008.

[7] S. Dolan. Fun with semirings: a functional pearl on the abuse of linear algebra. In *ACM SIGPLAN Notices*, volume 48, pages 101–110. ACM, 2013.

[8] D. Ellis. Lagrange Interpolator Polynomial. https://www.mathworks.com/matlabcentral/fileexchange/

13151-lagrange-interpolator-polynomial?focused=5083678&tab=function. [Dostopano 25. 7. 2017].

[9] Matlab funkcija nnmf. https://www.mathworks.com/help/stats/nnmf.html. [Dostopano 26. 8. 2017].

[10] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms*. Operations Research/Computer Science Interfaces Series. Springer US, 2008.

[11] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[12] S. Karaev and P. Miettinen. Cancer: Another algorithm for subtropical matrix factorization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 576–592. Springer, 2016.

[13] S. Karaev and P. Miettinen. Capricorn: An algorithm for subtropical matrix factorization. In *16th SIAM International Conference on Data Mining*. SIAM, 2016.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems, 2009.

[15] P. Miettinen and J. Vreeken. Model order selection for boolean matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 51–59. ACM, 2011.

[16] B. A. Pearlmutter and H. Šmigoc. Nonnegative factorization of a data matrix as a motivational example for basic linear algebra. arXiv:1706.09699.

[17] Sodo sistemski operater Distribucijskega Omrežja z električno energijo, d.o.o. https://www.sodo.si/kdo-smo. [Dostopano 6.8.2017].

[18] A. Veber Horvat, M. Miklavčič, and D. Bokal. Implementacija procesa za napovedovanje proizvodnje razpršenih virov električne energije v distribucijskem omrežju. In *CIRED - Mednarodni kongres za distribucijske elektroenergetske sisteme*. CIRED, 2017.

[19] E. W. Weisstein. "Frobenius Norm." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/FrobeniusNorm.html`. [Dostopano 31.7.2017].

[20] Fakulteta za matematiko in fiziko. `https://www.fmf.uni-lj.si/en/`. [Dostopano 3. 9. 2017].

[21] Fakulteta za računalništvo in informatiko. `https://www.fri.uni-lj.si/en`. [Dostopano 3. 9. 2017].